21世纪高等学校计算机类课程创新规划教材 · 微课版

MySQL 数据库应用与开发

姜桂洪 主编







教学
资源



清华大学出版社

21 世纪高等学校计算机类课程创新规划教材 · 微课版

MySQL 数据库应用与开发

姜桂洪　主编
孙福振　苏　晶　编著

清华大学出版社
北　京

内 容 简 介

本书采用 MySQL 5.7.17 版本软件，全面系统地讲述了 MySQL 数据库的基础知识和基本操作，以及各种常用数据库对象的创建和管理、MySQL 语言及其应用、数据库的备份与恢复、安全管理、日志管理与性能优化等。对数据操作中较为常用的数据检索、数据完整性、视图、存储过程、触发器、并发控制等内容进行了详细的阐述，并介绍了利用 PHP 访问 MySQL 数据库的方法和利用 JSP 开发 MySQL 数据库应用系统的基本过程。

全书体系完整、结构安排合理、内容翔实、例题丰富、可操作性强，并对主要操作单元配制微课视频。内容涵盖了 MySQL 数据库要用到的主要知识点。

本书适合作为高等院校本科、专科计算机及相关专业数据库管理和应用系统开发课程的教材，也可作为从事数据库管理与开发的信息技术领域的科技工作者的参考用书。另外，本书还配有辅导教材《MySQL 数据库应用与开发习题解答与上机指导》，以帮助读者进一步巩固所学 MySQL 数据库的知识。

图书在版编目（CIP）数据

MySQL 数据库应用与开发/姜桂洪主编. —北京：清华大学出版社，2018
（21 世纪高等学校计算机类课程创新规划教材 · 微课版）
ISBN 978-7-302-49592-5

Ⅰ. ①M…　Ⅱ. ①姜…　Ⅲ. ①SQL 语言－程序设计　Ⅳ. ①TP311.132.3

中国版本图书馆 CIP 数据核字（2018）第 030372 号

责任编辑：魏江江　薛　阳
封面设计：刘　键
责任校对：时翠兰
责任印制：李红英

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 装 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　**印　张：**22　　**字　　数：**537 千字
版　　次：2018 年 7 月第 1 版　　**印　　次：**2018 年 7 月第1 次印刷
印　　数：1～1500
定　　价：59.80 元

产品编号：076087-01

前　　言

Oracle 公司的 MySQL 是目前最流行的关系数据库管理系统之一。MySQL 所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 数据库以其精巧灵活、运行速度快、经济适用性强、开放源码等优势，作为网站数据库获得许多中小型网站的开发公司的青睐。MySQL 性能卓越，搭配 PHP 和 Apache 可组成良好的软件开发环境，并且已经大量部署到中小型企业和高校的教学平台。

本书从教学实际需求出发，结合初学者的认知规律，由浅入深、循序渐进地讲解 MySQL 数据库管理与开发过程中的知识。全书以 MySQL 数据库软件和数据库对象的基本操作为主线，将数据库理论内容嵌入到实际操作中去介绍，能够让学生在操作过程中进一步认知数据管理的理念，体察数据操作的优势，提高数据处理的能力。

全书体系完整、可操作性强，以大量的例题对常用知识点操作进行示范，所有的例题全部通过调试，内容涵盖了设计一个数据库应用系统要用到的主要知识。重点操作还增加了 Workbench 软件可视化操作的详细过程，并对主要操作单元配制 100 多个微课视频。

本书共分 15 章，现将本书的主要内容简单介绍如下：

第 1 章　MySQL 数据库概述。介绍有关 MySQL 数据库管理系统的基础知识和关系数据库理论。

第 2 章　MySQL 语言基础。介绍 MySQL 的数据类型、运算符、常用函数和表达式等。

第 3 章　MySQL 数据库的基本操作。介绍 MySQL 数据库的设计、创建和管理的基本操作，以及利用 MySQL Workbench 管理数据库的基本操作等内容。

第 4 章　表及数据完整性。介绍 MySQL 数据表的创建和管理、数据的常用操作和数据完整性的实现等内容。

第 5 章　数据检索。介绍利用 select 语句进行数据查询的内容，包括单表查询、多表连接、子查询及使用正则表达式进行模糊查询等。

第 6 章　索引和视图。介绍索引和视图的创建及管理，以及视图的应用等。

第 7 章　MySQL 编程基础。主要介绍变量、begin…end 语句块的应用，自定义函数创建和维护管理，MySQL 的控制流语句的应用。

第 8 章　存储过程、游标和触发器。通过介绍存储过程的创建、应用和管理，并利用存储过程实现了游标、触发器和事件等数据库对象的创建及应用。

第 9 章　并发事务与锁机制。介绍事务的并发处理机制和锁机制的功能和应用。

第 10 章　权限管理及安全控制。介绍 MySQL 权限系统的工作原理、账户管理、权限管理等 MySQL 数据库安全常见问题。

第 11 章　备份与恢复。介绍 MySQL 数据库的备份和恢复的基本理论和基本操作，还

介绍表的导入与导出等基本操作。

第 12 章 MySQL 性能优化。介绍优化 MySQL 服务器的方法、优化查询的概念和操作。

第 13 章 MySQL 日志文件管理。分别介绍错误日志、二进制日志、通用查询日志和慢查询日志的文件管理和应用。

第 14 章 使用 PHP 操作 MySQL 数据库。介绍 PHP 语言的特点和搭建 PHP+MySQL 的集成开发环境的过程,以及使用 PHP 操作 MySQL 数据库的常见方法。

第 15 章 基于 JSP 技术的 MySQL 数据库应用开发实例。介绍基于 JSP 技术的 MySQL 数据库应用开发实例的数据库设计、在线考试系统的应用开发、运行与测试过程。

本书由姜桂洪、孙福振、苏晶等编写,由姜桂洪统稿。在本书编写过程中还参阅了大量的数据库方面的文献和网站资料,在此对提供者一并深表感谢。

另外,本书还配有辅导教材《MySQL 数据库应用与开发习题解答与上机指导》(姜桂洪等编著),内容包括本书所有习题的详尽参考答案、模拟试题、MySQL 软件安装配置的常见操作、MySQL 数据库的常用可视化软件 Workbench、Navicat 和 phpMyAdmin 的安装、配置和基本操作方法,以及按本书章节顺序配备的实验及实验指导等。

由于作者水平有限,书中纰漏之处在所难免,恳请读者批评指正。

编　者

2018 年 1 月

目　录

第 1 章　MySQL 数据库概述

数据库技术是计算机科学的重要组成部分，也是信息管理的技术依托，主要用于研究如何向用户提供具有共享性、安全性和可靠性数据的方法。数据库技术解决了计算机信息处理过程中有效地组织和存储海量数据的问题。而大数据的发展更是将数据库技术的应用平台推上一个新的高度。数据库的建设规模、数据信息的存储容量和处理能力已成为衡量一个国家现代化程度的重要标志。

具体来说，数据库技术包括数据库系统、SQL 语言、数据库访问技术等。像 MySQL、Oracle、SQL Server 和 DB2 等都是目前常用的数据库管理系统软件。尤其是 MySQL 已经成为目前软件行业市场份额提高最快的数据库软件。

本章主要介绍数据库系统的有关概念以及 MySQL 数据库的基本知识。

1.1　认识 MySQL 数据库

MySQL 是一个开放源码的小型、跨平台数据库管理系统，被广泛地应用在 Internet 上的中小型网站中。目前 MySQL 和 Oracle 数据库一样，都属于甲骨文公司。由于其具有体积小、运行速度快、总体拥有成本低、开放源码的优势，许多中小型网站都为了降低网站总体拥有成本而选择了 MySQL 作为网站数据库。下面介绍与 MySQL 数据库相关的基本知识。

1. MySQL 数据库的发展背景

MySQL 是由一个天赋极高的程序员 Monty Widenius 经过近二十年的坚持而成功开发的数据库软件。

1996 年发布了能够在小范围内使用的 MySQL 1.0 版。

1999 年 MySQL AB 公司在瑞典成立。Monty 与 Sleepycat 公司合作开发出 Berkeley DB(简称为 BDB)引擎，由于 BDB 支持事务处理，MySQL 数据库从此能够支持事务处理了。

2000 年 MySQL 数据库集成了存储引擎 InnoDB，这个引擎同样支持事务处理，还支持行级锁。该引擎之后被证明是最为成功的 MySQL 事务存储引擎。

2003 年 12 月，MySQL 5.0 版本发布，提供了视图和存储过程等功能。

2008 年 1 月，MySQL AB 公司被 Sun 公司收购，Sun 公司对其进行了大量的推广、优化和 bug 修复等工作。

2008 年 11 月，MySQL 5.1 发布，它提供了分区、事件管理，以及基于行的复制和基于磁盘的 NDB 集群系统，同时修复了大量的 bug。

2009 年 4 月，甲骨文公司收购 Sun 公司，自此 MySQL 数据库进入 Oracle 时代。2010

年 12 月，MySQL 5.5 发布，其主要新特性包括半同步的复制及对 SIGNAL/RESIGNAL 的异常处理功能的支持，最重要的是 InnoDB 存储引擎终于变为当前 MySQL 数据库的默认存储引擎。

2013 年 2 月，甲骨文公司宣布 MySQL 5.6 正式版发布，首个正式版本号为 5.6.10。2013 年 4 月发布 MySQL 5.7.1 版本。

本书采用 2016 年 12 月发布的 mySQL 5.7.17 版本编写。

2. MySQL 使用优势

MySQL 数据库的应用非常广泛，尤其是在 Web 应用方面。因此，MySQL 数据库的市场份额迅速增长。许多大型网站之所以选择使用 MySQL 数据库来存储数据，主要是因为 MySQL 数据库具有以下 4 个方面的优势。

(1) MySQL 数据库是开放源代码的数据库。任何人都可以获取 MySQL 数据库的源代码，可以修改 MySQL 数据库的缺陷，并以任何目的来使用该数据库。MySQL 数据库作为一款自由的软件，完全继承了自由软件基金会(GNU)的思想，这保证了 MySQL 数据库是一款可以自由使用的数据库。

(2) MySQL 数据库的跨平台性。MySQL 不仅可以在 Windows 系列的操作系统上运行，还可以在 UNIX、Linux 和 Mac OS 等操作系统上运行。许多网站都选择 UNIX 和 Linux 作为网站的服务器，因此 MySQL 数据库的跨平台性保证其在 Web 应用方面的优势。Microsoft 公司的 SQL Server 数据库是一款非常优秀的商业数据库，尤其是其可视化平台的操作是一个突出的优势，但只能用在 Windows 操作系统上。

(3) MySQL 的价格优势。MySQL 数据库是一款自由软件，社区版本的 MySQL 数据库软件都是免费使用的，即使是需要付费的附加功能，其价格也是很便宜的。相对于 Oracle、SQL Server 和 DB2 这些价格昂贵的商业软件，MySQL 数据库具有绝对的价格优势。

(4) 功能强大使用方便。MySQL 数据库是一个多用户、多线程的 SQL 数据库服务器，它是 C/S 结构的实现，由一个服务器守护程序 mysqlId 和很多不同的客户程序以及库组成。MySQL 数据库能够快速、有效和安全地处理大量的数据。相对于 Oracle 数据库来说，MySQL 数据库的使用是非常简单的。MySQL 能够快速、有效和安全地处理大量的数据，并达到快速、健壮和易用的目标。

3. MySQL 系统特性

MySQL 数据库市场份额的快速增长，除了自身的优势外，也离不开它的特性，具体优势可以描述如下：

- 使用 C 和 C++编写，并使用了多种编译器进行测试，保证源代码的可移植性。同时，为 PHP、Java、C、C++、Python、Perl、Eiffel、Ruby 和 Tcl 等多种编程语言提供了应用程序接口 API。
- 支持 Windows、Linux、Mac OS、AIX、FreeBSD、HP-UX、NovellNetware、OpenBSD、OS/2 Wrap 以及 Solaris 等多种操作系统。
- MySQL 支持多线程，能够充分利用 CPU 资源。
- 能够自动优化 SQL 查询算法，有效地提高了信息查询速度。
- 能够作为一个单独的应用程序应用在客户端—服务器网络环境中，也可以作为一个

库嵌入到其他软件中。

- 提供多种自然语言支持，常见的编码如中文的 GB 2312、BIG5 及国际通用转换格式 UTF-8 等都可以用作数据表名和数据列名。
- 提供 TCP/IP、ODBC 和 JDBC 等多种数据库连接技术。
- 支持多种存储引擎，提供用于管理、检查、优化数据库操作的管理工具。
- 具有大型数据库所有常用功能，可以处理拥有亿万条记录级的海量数据。

4. MySQL 发行版本

MySQL 数据库版本类型多，对于低于 MySQL 5.0 的版本，官方将不再提供支持。而所有发布的 MySQL(Current Generally Available Release，目前一般可用版本)版本已经经过严格标准的测试，可以保证其安全可靠地使用。

(1) 根据操作系统的类型来划分，大体上可以分为 Windows 版、UNIX 版、Linux 版和 Mac OS 版。因为 UNIX 和 Linux 操作系统下的版本也有很多，因此，不同的 UNIX 和 Linux 版本有对应的 MySQL 版本。如果要下载 MySQL 数据库，必须要了解自己使用的操作系统，然后根据操作系统来下载相应的 MySQL 数据库。

(2) 根据发布顺序来划分，MySQL 数据库可以分为 MySQL 4.0、MySQL 5.0、MySQL 5.1 以及 MySQL 5.7 等系列版本。MySQL 5.7 是目前最新开发的普遍常用(Generally Available，GA)的稳定系列版本，目前已经可以正常使用。MySQL 的命名机制由 3 个数字和 1 个后缀组成。

例如，MySQL 5.7.17 版本的含义如下：

① 第 1 个数字“5”是主版本号，描述了文件格式，即所有版本 5 的发行版都有相同的文件格式。

② 第 2 个数字“7”是发行级别，主版本号和发行级别组合在一起便构成了发行序列号。

③ 第 3 个数字“17”是该发行系列的版本号，随每次新发布版本递增。

(3) 根据 MySQL 数据库的开发情况，可将其分为 Alpha、Beta、Gamma 和 Generally Available 等版本。

Alpha：处于开发阶段的版本，可能会增加新的功能或进行重大修改。

Beta：处理测试阶段的版本，开发已经基本完成，但是没有进行全面的测试。

Gamma：该版本是发行过一段时间的 Beta 版，比 Beta 版要稳定一些。

Generally Available：该版本已经足够稳定，可以在软件开发中应用了。有些资料会将该版本称为 Production 版。

(4) 根据 MySQL 数据库用户群体的不同，将其分为社区版(Community Edition)和企业版(Enterprise)。社区版是自由下载而且是免费开源的，但是没有官方的技术支持。企业版提供了最全面的高级功能、管理工具和技术支持，实现了最高水平的 MySQL 数据库可扩展功能、安全性、可靠性和无故障运行时间。它可在开发、部署和管理关键业务型 MySQL 应用程序的过程中降低风险、减少成本和减少复杂性。企业版还能够以很高的性价比为企业提供数据仓库应用，支持事务处理，提供完整的提交、回滚、崩溃恢复和行级锁定功能。但是该版本需付费使用，官方提供电话技术支持。

5. MySQL 5.7 的新功能

MySQL 5.7 与以前的版本相比，主要包括以下几个方面的新功能。

(1) 支持 JSON。JSON(JavaScript Object Notation)是一种存储信息的格式,可以很好地替代 XML。从 MySQL 5.7.8 版本开始,MySQL 将支持 JSON,而在此版本之前,只能通过 strings 之类的通用形式来存储 JSON 文件,这样做的缺陷很明显,就是必须要自行确认和解析数据,解决更新中的困难,在执行插入操作时忍受较慢的速度。

(2) 性能和可扩展性。改进 InnoDB 的可扩展性和临时表的性能,从而实现更快的网络和大数据加载等操作。

(3) 改进复制以提高可用性的性能。改进复制包括多源复制、多线程增强、在线 GTIDs 和增强的半同步复制。

(4) 性能模式提供更好的视角。增加了许多新的监控功能,以减少空间和过载,使用新的 SYS 模式显著提高易用性。

(5) 保证数据库的安全。以安全第一为宗旨,提供了很多新的功能,从而保证数据库的安全。

(6) 对多种性能进行了优化。重写了大部分解析器、优化器和成本模型,这提高了可维护性、可扩展性和性能。

(7) 支持 GIS(Geographic Information System,地理信息系统)。MySQL 5.7 全新的功能包括 InnoDB 空间索引和 Boost 几何,同时提高完整性和标准符合性。

1.2 数据库的基本概念

数据库技术经过长期的发展已经形成了系统的科学理论,数据管理和信息处理是数据库技术的主要内容。

1.2.1 信息与数据库

1. 数据和信息

数据(Data)是描述事物的符号记录,它有多种表现形式,可以是文本、图表、图形、图像、声音、语言、视频等。

信息(Information)是具有特定意义的数据。信息不仅具有能够感知、存储、加工、传播和再生等自然属性,同时也是具有重要价值的社会资源。信息是用一定的规则或算法筛选的数据集合。

2. 数据库

数据库(Database,DB)是长期存储在计算机内、有组织、可共享的大量数据的集合。数据库中的数据需要创建数据模型来描述,如网络、层次、关系模型。在数据库中的数据具有冗余度小、独立性高和易扩展的特点。

例如,可以利用 MySQL 软件创建一个教务管理数据库 teaching,将学生的基本信息(学号、姓名、性别、出生日期、手机号等)存放在一起,就可以创建 teaching 数据库中的一个学生信息表 student,如表 1-1 所示。将学生成绩信息(学号、课程号、平时成绩、期末成绩)存放在一起,就可以创建 teaching 数据库中的学生成绩表 score,如表 1-2 所示。

数据库中的数据除了其本身外,还包含着数据库对数据的语义描述。例如,表 1-1 中数据 18137221508 经过 studentno 语义描述,就成为学号,而数据 13198765431 经过 phone 语

义描述就成为一个手机号。而数据不经过语义描述，其本身的意义不完整，只表示一个常量值。

表 1-1　student

studentno	sname	sex	birthdate	phone
17112345678	高伟业	男	1999-09-09	13198765431
18137221508	曲梅影	女	2000-12-12	18278965439
…	…	…	…	…

表 1-2　score

studentno	courseno	daily	final
17112345678	c06108	89	98
18137221508	c05109	95	96
…	…	…	…

1.2.2　结构化查询语言(SQL)

SQL(Structured Query Language，结构化查询语言)是用于管理数据的一种数据库查询和程序设计语言。其主要用于存取、查询和更新数据，还能够管理关系数据库系统的数据库对象。

SQL 现在有许多不同的类型，有三个主要的标准：ANSI(美国国家标准机构)SQL，对 ANSI SQL 修改后在 1992 年采纳的标准，称为 SQL-92 或 SQL2。最近的 SQL-99 标准，从 SQL2 扩充而来并增加了对象关系特征和许多其他新功能。其次，各大数据库厂商提供不同版本的 SQL，这些版本的 SQL 不但能包括原始的 ANSI 标准，而且在很大程度上支持 SQL-92 标准。

1. SQL 语言特点

(1) 一体化：SQL 集数据定义 DDL、数据操纵 DML 和数据控制 DCL 于一体，可以完成数据库中的全部工作。

(2) 使用方式灵活：它具有两种使用方式，既可以直接以命令方式交互使用；也可以嵌入使用，嵌入到 C、C++、FORTRAN、COBOL 和 Java 等主语言中使用。

(3) 非过程化：只需要提供操作要求，不必描述操作步骤，也不需要导航。使用时只需要告诉计算机"做什么"，而不需要告诉它"怎么做"。

(4) 语言简洁，语法简单，好学好用：在 ANSI 标准中，只包含了 94 个英文单词，核心功能只用 6 个动词，语法接近英语口语。

2. SQL 查询语言的组成

结构化查询语言包含如下几个部分。

(1) 数据定义语言(Data Definition Language，DDL)：其语句包括动词 create、alter 和 drop。在数据库中创建、修改或删除数据库对象，如表、索引、视图、存储过程、触发器、事件等。

（2）数据操作语言（Data Manipulation Language，DML）：包括动词 select、insert、update 和 delete。它们分别用于查询、插入、修改和删除表中的数据行等。select 是用得最多的动词，也称为数据查询语言（Data Query Language，DQL），其他 DQL 常用的保留字有 where、order by、group by 和 having。这些 DQL 保留字常与其他类型的 SQL 语句一起使用。

（3）数据控制语言（Data Control Language，DCL）：包括 grant 语句和 revoke 等语句。通过 grant 语句获得权限许可，revoke 可以撤销权限许可，确定单个用户和用户组对数据库对象的访问权限。某些数据库管理系统可用 grant 或 revoke 控制对表单个列的访问。

（4）事务处理语言（Transaction Processing Language，TPL）：它的语句能确保被 DML 语句影响的表的所有行及时得以更新。TPL 语句包括 begin transaction、commit 和 rollback。

（5）指针控制语言（Pointer Control Language，CCL）：它的语句，如 declare cursor、fetch into 和 update where current 用于对一个或多个表单独行的操作。

在应用程序中，也可以通过 SQL 语句来操作数据。例如，可以在 Java 语言中嵌入 SQL 语句。通过执行 Java 语言来调用 SQL 语句，这样即可在数据库中插入数据、查询数据。SQL 语句也可以嵌入到 C#、PHP 等编程语言中。

1.2.3 数据库管理系统

数据库管理系统（Database Management System，DBMS）位于用户和操作系统之间，是一种操纵和管理数据库的大型软件，用于建立、使用和维护数据库。DBMS 可以对数据库进行统一的管理和控制，以保证数据的安全性和完整性，是数据库系统的核心。数据库中数据的插入、修改和检索均要通过数据库管理系统进行。

用户通过 DBMS 访问数据库中的数据，数据库管理员（Database Administrator，DBA）也通过 DBMS 进行数据库的维护工作。它可使多个应用程序和用户采用不同的方法在同一时刻或不同时刻去建立、修改和查询数据库。

如图 1-1 所示，DBMS 提供了数据定义语言（DDL）、数据操作语言（DML）和应用程序，可以提供用户定义数据库的模式结构与权限约束，实现对数据的追加、删除等操作。数据库管理系统是由多种不同的程序模块组成的，基本数据库管理系统的系统架构包括 4 部分。

（1）存储管理（Storage Manager）。数据库管理系统通常会自行配置磁盘空间，将数据存入存储装置的数据库。

（2）查询处理（Query Processor）。负责处理用户下达的查询语言命令语句，可以再细分成多个模块负责检查语法、优化查询命令的处理程序。

（3）事务管理（Transaction Manager）。负责处理数据库的事务，保障数据库商业事务的操作需要，以及并发控制管理（Concurrency-Control Manager）资源锁定等。

（4）恢复管理（Recovery Manager）。恢复管理主要是日志管理（Log Manager），负责记录数据库的所有操作，可以恢复数据库系统存储的数据到指定的时间点。

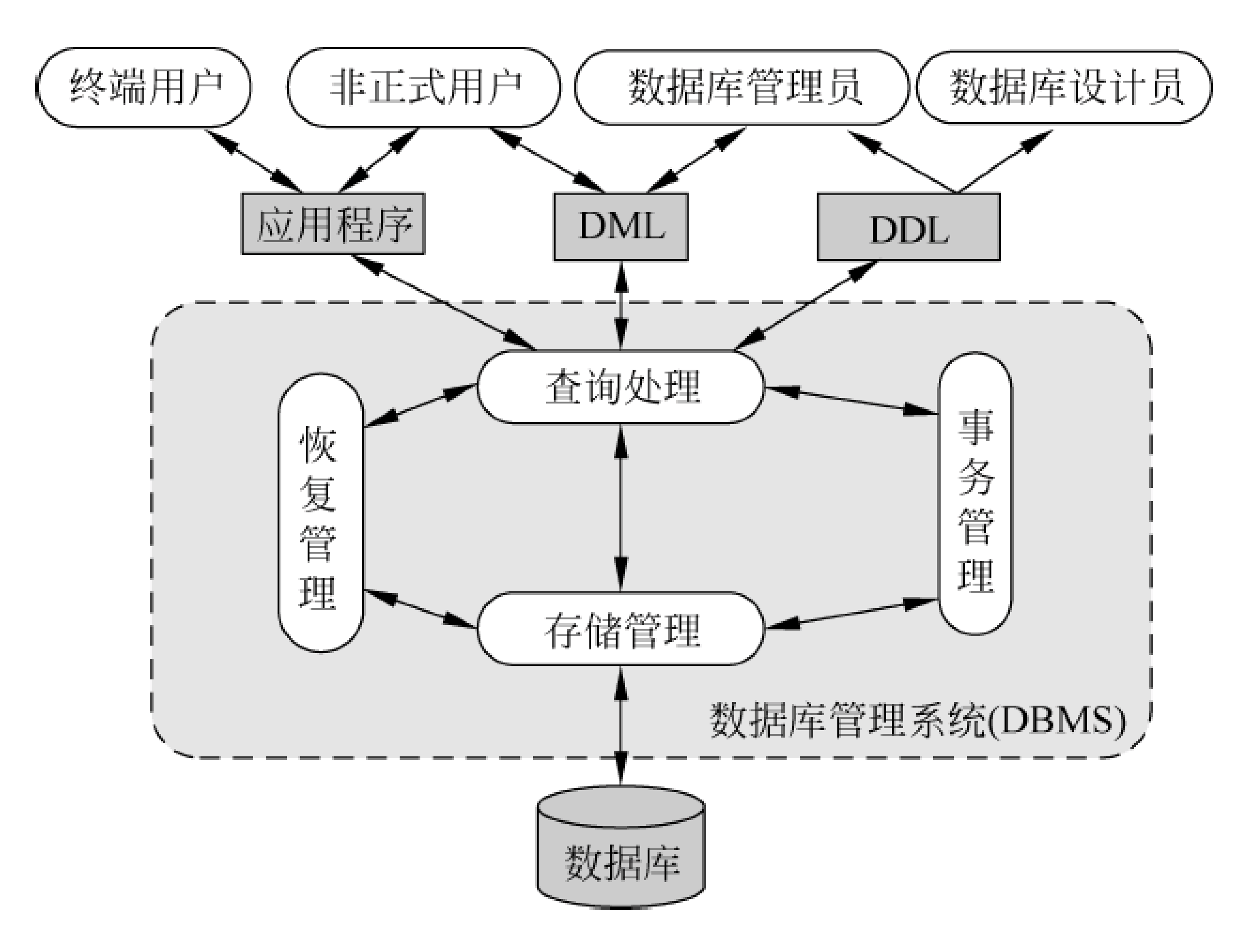


图 1-1　数据库管理系统架构示意图

1.2.4　数据库系统

1. 数据库系统的组成

数据库系统(Database System,DBS)通常由硬件、软件、数据库和用户组成,管理的对象是数据。其中软件主要包括操作系统、各种宿主语言、实用程序以及数据库管理系统。数据库系统的架构如图 1-2 所示,数据库系统包括四大组件:用户、数据、软件和硬件。

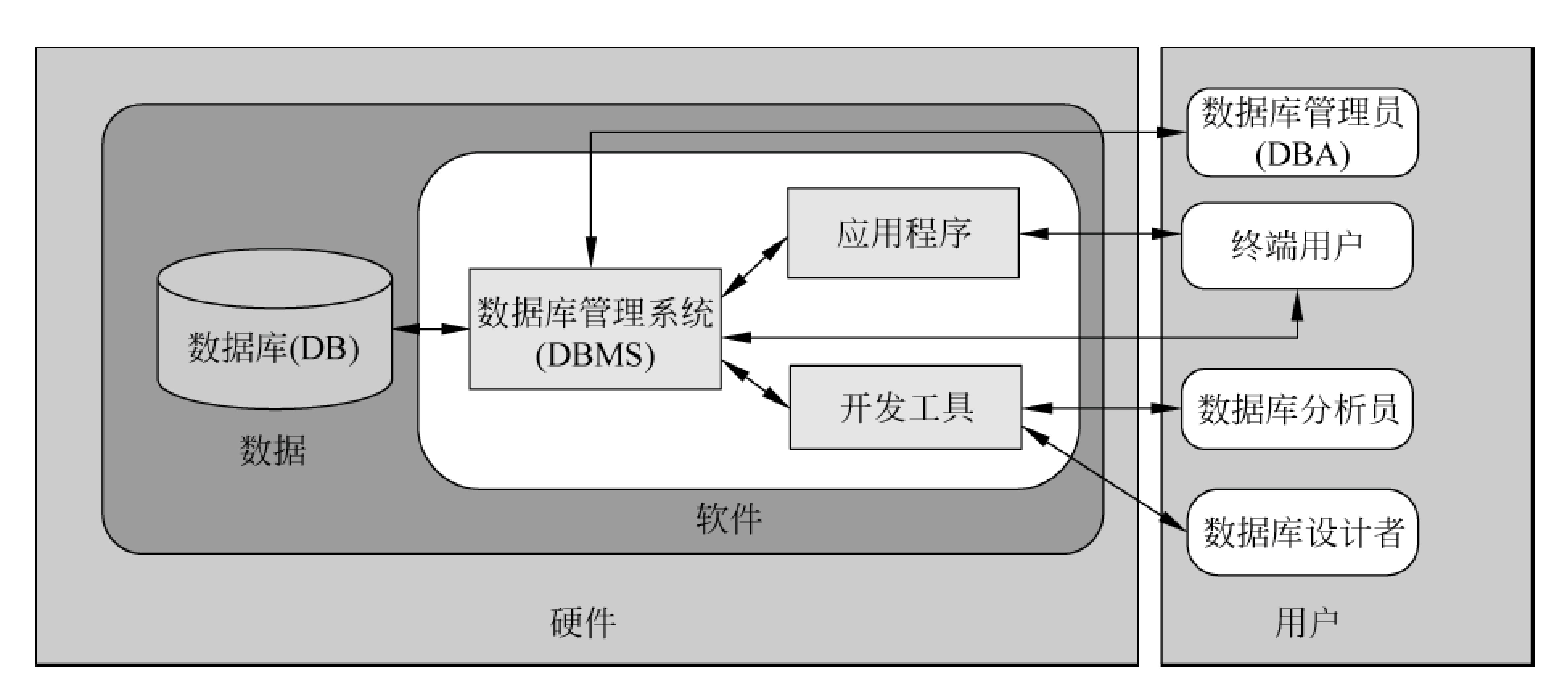


图 1-2　数据库系统架构

(1) 用户(Users)。用户执行 DDL 定义数据库架构,使用 DML 新增、删除、更新和查询数据库的数据,通过操作系统访问数据库的数据。按不同角色划分,用户可以分为多种,如终端用户(End-Users)、数据库设计者(Database Designers)、系统分析师(System Analyst,SA)、应用程序设计师(Application Programmer)和数据库管理员等。数据库管理员负责创建、监控和维护整个数据库,一般是由业务水平较高、资历较深的人员担任。

(2) 数据(Data)。数据库系统中的数据种类包括永久性数据(Persistent Data)、索引数据(Indexes)、数据字典(Data Dictionary)和事务日志(Transaction Log)等。

(3) 软件(Software)。指在数据库环境中使用的软件,包括数据库管理系统、应用程序和开发工具(Development Tools)等。

(4) 硬件(Hardware)。安装数据库相关软件的硬件设备,包含主机(CPU、内存和网卡等)、磁盘阵列、光驱和备份装置等。

2. 数据库系统的体系结构

数据库系统的体系结构主要包括如下几种结构:集中式、客户-服务器式(Client/Server,C/S)、浏览器-服务器式(Browser/Server,B/S)和分布式等。

(1) 集中式结构。集中式系统是运行在一台计算机上,不与其他计算机系统交互的数据库系统,例如运行在个人计算机上的单用户数据库系统和运行在大型主机上的高性能数据库系统。

(2) C/S 结构。C/S 结构可将数据库功能大致分为前台客户端系统和后台服务器系统。客户端系统主要包括图形用户界面工具、表格及报表生成和书写工具等;服务器系统负责数据的存取和控制,包括故障恢复和并发控制等。客户机通过网络将要求传递给服务器,服务器按照客户机的要求返回结果。

(3) B/S 结构。B/S 结构将客户机上的应用层从客户机中分离出来,集中于一台高性能的计算机上,成为应用服务器,也称为 Web 服务器。这种模式统一了客户端,将系统功能实现的核心部分集中到服务器上,简化了系统的开发、维护和使用。客户机上只要安装一个浏览器,服务器安装 SQL Server、Oracle 等数据库。Web 服务器充当了客户端与数据库服务器的中介,架起了用户界面与数据库之间的桥梁。

(4) 分布式结构。分布式数据库系统是计算机网络发展的必然产物,分布式数据库系统由多台计算机组成,每台计算机都配有各自的本地数据库。在分布式数据库系统中,大多数处理任务由本地计算机访问本地数据库完成局部应用。该系统满足了地理上分散组织对于数据库应用的需求。该系统通常由计算机网络连接起来,被连接的逻辑单位(包括计算机、外部设备等)称为节点。对于少量本地计算机不能胜任的处理任务,可以通过网络同时存取和处理多个异地数据库中的数据。

1.3 关系数据库理论

关系数据库(Relational Database,RDB)是基于关系模型的数据库,是应用数学理论处理和组织数据的一种方法。

1.3.1 概念模型及其表示方法

概念模型是现实世界信息的抽象反映,不依赖于具体的计算机系统,是现实世界到计算机世界的一个中间层次。

1. 实体的相关概念

(1) 实体(Entity)。客观存在并可以相互区分的事物叫实体。从具体的人、物、事件到抽象的状态与概念都可以用实体抽象地表示。例如,在学校里,一名学生、一名教师、一门课程等都称为实体。

(2) 属性(Attribute)。属性是实体所具有的某些特性,通过属性对实体进行描述。实

体是由属性组成的。一个实体本身具有许多属性，能够唯一标识实体的属性称为该实体的主键。例如，学号是学生实体的主键，每个学生都有一个属于自己的学号，通过学号可以唯一确定是哪位学生，在同一个学校里，不允许有两个学生具有相同的学号。

(3) 主键(Primary Key)。一个实体往往有多个属性，这些属性之间是有关系的，它们构成该实体的属性集合。如果其中有一个属性或者多个属性构成的子集能够唯一标识整个属性集合，则称该属性子集为属性集合的主键。

(4) 实体型(Entity Type)。具有相同属性的实体必然具有共同的特征和性质。用实体名及其属性名集合来抽象和刻画同类实体，称为实体型。例如，学生(学号，姓名，性别，出生日期，班级，入学成绩)就是一个实体型。

(5) 实体集(Entity Set)。同型实体的集合称为实体集。例如，全体学生就是一个实体集。

(6) 联系(Relationship)。现实世界的事物之间是有联系的。这些联系必然要在信息世界中加以反映。例如，教师实体与学生实体之间存在着教和学的联系。

2. 实体间的联系

实体间的联系是错综复杂的，但就两个实体型的联系来说，如图 1-3 所示，主要有以下三种类型。

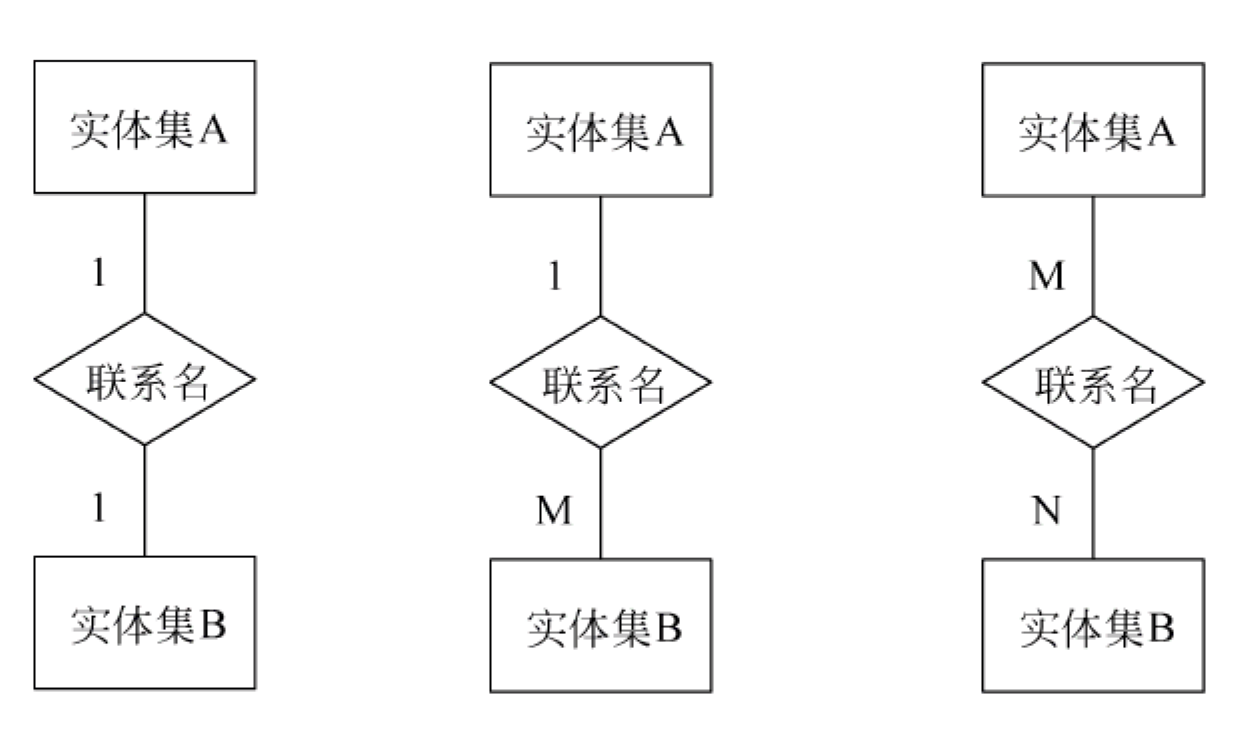


图 1-3　两个实体集之间的联系

(1) 一对一的联系(1∶1)。对于实体集 A 中的每一个实体，实体集 B 中至多有一个实体与之联系，反之亦然，则称实体集 A 与实体集 B 具有一对一联系，记为 1∶1。例如，通常一个班内都只有一个班长，班级和班长之间具有一对一联系。

(2) 一对多联系(1∶M)。对于实体集 A 中的每一个实体，实体集 B 中有 M 个实体(M≥2)与之联系；反过来，对于实体集 B 中的每一个实体，实体集 A 中至多有一个实体与之联系，则称实体集 A 与实体集 B 具有一对多联系，记为 1∶M。例如，一个班内有多名同学一名同学只能属于一个班，即班级与同学之间具有一对多联系。

(3) 多对多联系(M∶N)。对于实体集 A 中的每一个实体，实体集 B 中有 N 个实体(N≥0)与之联系；反过来，对于实体集 B 中的每一个实体，实体集 A 中也有 M 个实体(M≥0)与之联系，则称实体集 A 与实体集 B 具有多对多联系，记为 M∶N。例如，学生在选课时，一个学生可以选多门课程，一门课程也可以被多个学生选取，则学生和课程之间具有多对多联系。

3. 概念模型的表示方法

概念模型的表示方法很多，其中最常用的是实体-联系模型(Entity-Relationship Model)，简称为 E-R 模型。在 E-R 概念模型中，信息由实体型、实体属性和实体间的联系三种概念单元来表示。

(1) 实体型表示建立概念模型的对象，用长方框表示，在框内写上实体名。如学生，课程等。

(2) 实体属性是实体的说明。用椭圆框表示实体的属性，并用无向边把实体与其属性连接起来。例如，学生实体有学号、姓名、性别、出生日期、手机号等属性。

(3) 实体间的联系是两个或两个以上实体类型之间的有名称的关联。实体间的联系用菱形框表示，菱形框内要有联系名，并用无向边把菱形框分别与有关实体相连接，在无向边的旁边标上联系的类型。例如，可以用 E-R 图来表示某学校学生选课情况的概念模型，如图 1-4 所示。一个学生可以选修多门课程，一门课程也可以被多个学生选修，因此，学生和课程之间具有多对多的联系。

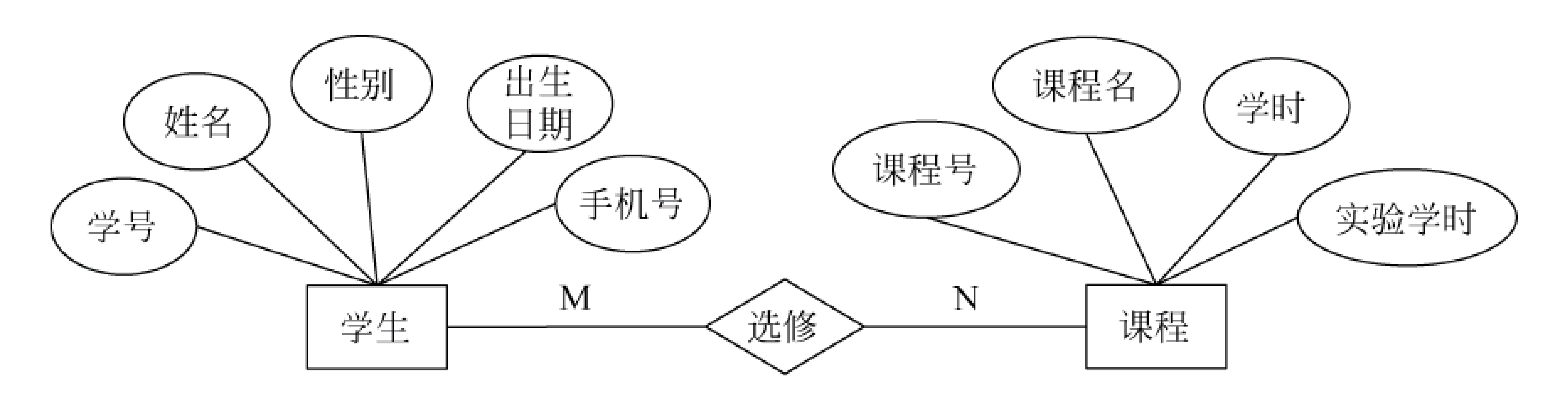


图 1-4　实体、实体属性及实体联系模型

1.3.2　数据模型

在概念模型基础上建立的适用于数据库层的模型，称为数据模型。数据模型能够精确地描述系统的静态特征、动态特征和完整性约束条件。

1. 数据模型的三要素

数据模型由数据结构、数据操作和完整性约束三个要素组成。

(1) 数据结构。数据结构是对象和对象间联系的表达和实现，是所研究的对象类型的集合，用于描述数据库系统的静态特性。数据结构所研究的是数据本身的类型、内容和性质，以及数据之间的关系。例如关系模型中的主键、外键等。

(2) 数据操作。数据操作用于描述数据库系统的动态特征，是对数据库中对象实例允许执行的操作集合，主要指检索和更新(插入、删除、修改)两类操作。数据模型必须定义这些操作的确切含义、操作符号、操作规则(如优先级)以及实现操作的语言。

(3) 完整性约束条件。数据完整性约束是一组完整性规则的集合，它规定数据库状态及状态变化所应满足的条件，以保证数据的正确性、有效性和相容性。完整性规则是给定的数据模型中数据及其联系所具有的制约和存储规则，用以限定符合数据模型的数据库状态以及状态的变化，以保证数据的正确、有效和相容。在关系模型中，一般关系必须满足实体完整性和参照完整性两个条件。

2. 常用数据模型

(1) 层次模型(Hierarchical Model)。层次数据库用树状结构表示实体之间联系的模型

称为层次模型，它的数据结构类似一棵倒置的树，每个节点表示一个记录类型，记录之间的联系是一对多的联系，现实世界中很多事物是按层次组织起来的。

层次模型的优点是结构清晰，表示各节点之间的联系简单；容易表示现实世界的层次结构的事物及其之间的联系。缺点是不能表示两个以上实体之间的复杂联系和实体之间的多对多联系；严格的层次顺序使数据插入和删除操作变得复杂。

(2) 网状模型(Network Model)。网状数据库是用来处理以记录类型为节点的网状数据模型的数据库。网状模型采用网状结构表示实体及其之间的联系。网状结构的每一个节点代表一个记录类型，记录类型可包含若干字段，联系用链接指针表示，去掉了层次模型的限制。由于网状模型比较复杂，一般实际的网状数据库管理系统对网状都有一些具体的限制。

网状模型的优点是能够表示实体之间的多种复杂联系。缺点是网状模型比较复杂，需要程序员熟悉数据库的逻辑结构；在重新组织数据库时，容易失去数据独立性。

(3) 关系模型(Relational Model)。关系数据库是目前流行的数据库。它是建立在关系数据库模型基础上的数据库，借助于集合代数等概念和方法来处理数据库中的数据，是用户看到的二维表格集合形式的数据库。关系模型是目前最重要的一种数据模型，关系数据库系统采用关系模型作为数据的组织方式，MySQL 数据库就是基于关系模型建立的。

(4) 面向对象模型(Object Oriented Model)。面向对象模型采用面向对象的方法来设计数据库。面向对象的数据库存储对象是以对象为单位，每个对象包含对象的属性和方法，具有类和继承等特点。Computer Associates 的 Jasmine 就是面向对象模型的数据库系统。

1.3.3 关系运算

关系数据操作就是关系运算，即从一个关系中找出所需要的数据。

1. 关系模型中的基本运算

在关系中访问所需的数据时，需要对关系进行一定的关系运算。关系数据库主要支持选择、投影和连接关系运算，它们源于关系代数中并、交、差、选择、投影和连接等运算。

(1) 选择。从一个表中找出满足指定条件的记录行形成一个新表的操作称为选择。选择是从行的角度进行运算得到新的表，新表的关系模式不变，其记录是原表的一个子集。选择关系运算如图 1-5 所示。

例如，在 student 关系中查询所有性别 sex 为“女”的学生。

(2) 投影。从一个表中找出若干字段形成一个新表的操作称为投影。投影是从列的角度进行的运算，通过对表中的字段进行选择或重组，得到新的表。新表的关系模式所包含的字段个数一般比原表少，或者字段的排列顺序与原表不同，其内容是原表的一个子集。投影关系运算如图 1-6 所示。

例如，在 student 关系中查询所有学生的学号 studentno 和出生日期 birthdate。

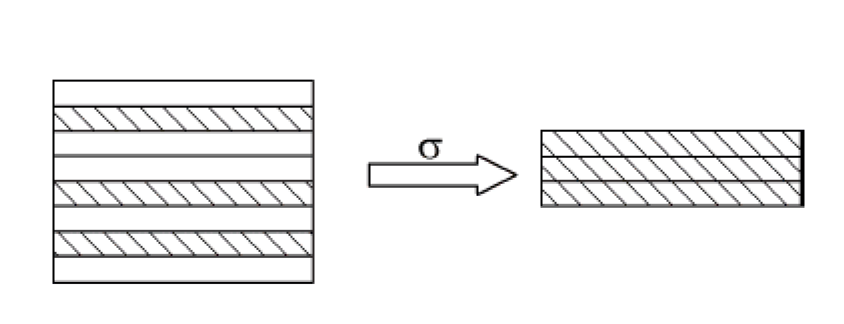


图 1-5　选择关系

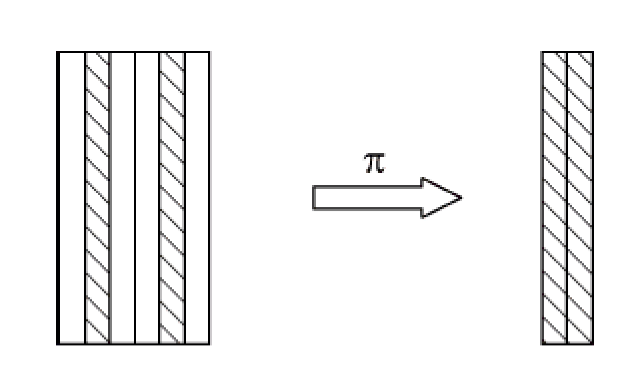


图 1-6　投影关系

(3) 连接。选择和投影都是对单表进行的运算。在通常情况下，需要从两个表中选择满足条件的记录。连接就是这样的运算方式，它是将两个表中的行按一定的条件横向结合，形成一个新的表。连接关系运算如图 1-7 所示。

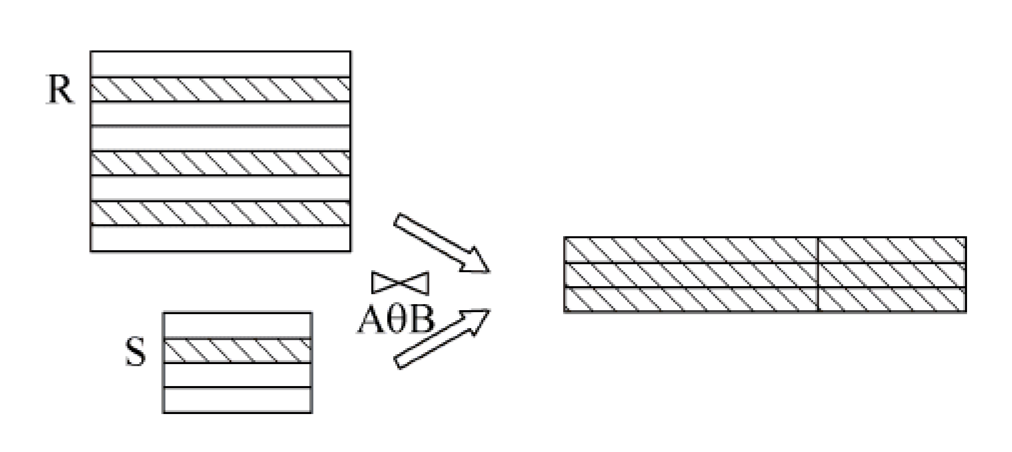


图 1-7　连接关系

例如，查询学生的姓名 sname 和期末成绩 final，两个数据项分别来自 student 关系和 score 关系，需要在两个关系连接之后，再从中按照一定条件筛选出 sname 和 final 的数据。

2. 关系模型的规范化设计

数据依赖是一个关系内部属性与属性之间的一种约束关系。这种约束关系是通过属性间值的相等与否体现出来的数据间的相关联系，它是现实世界属性间相互联系的抽象，是数据内在的性质，是语义的体现。

在数据库设计时，有一些专门的规则，称为数据库的设计范式，遵守这些规则，将创建设计良好的数据库，下面将逐一讲解数据库设计中的著名的范式理论。

(1) 第一范式(1NF)。第一范式的目标是确保每列的原子性。如果每列都是不可再分的最小数据单元(也称为最小的原子单元)，则满足第一范式(1NF)。

(2) 第二范式(2NF)。第二范式是在第一范式的基础上，要求确保表中的每列都和主码相关，即每一个非主属性都要完全函数依赖于主码。

(3) 第三范式(3NF)。第三范式是在第二范式的基础上，要求确保表中的每列都和码直接相关，而不是间接相关。如果一个关系满足 2NF，并且除了码以外的其他列都不相互依赖，则满足第三范式(3NF)。

在实际的数据库设计过程中，在利用规范化设计考查关系模式时，可以针对不同的关系以及关系转换成的表可以预估的数据量、物理存取路径等因素，对规范化关系采用一些另外的处理方法。

3. 关系的数据完整性

确保持久化数据检索不出错对于数据管理来说非常关键，也是数据库面临的最主要问题。没有数据完整性，则不能保证查询结果的正确性，那么可用性也就无从谈起了。

(1) 实体完整性。实体完整性是指关系的主关键字不能取“空值”。一个关系对应现实世界中的一个实体集。现实世界中的实体是可以相互区分、相互识别的，即它们应具有某种唯一性标识。在关系模式中，以主关键字作为唯一性标识，而主关键字中的属性(称为主属性)不能取空值，否则，表明关系模式中存在着不可标识的实体(因为空值是“不确定”的)。这与现实世界的实际情况相矛盾，这样的实体就不是一个完整实体。按照实体完整性规则要求，主属性不得取空值，如果主关键字是多个属性的组合，那么所有主属性均不得取空值。

例如，表 1-1 中的 studentno 作为主关键字，该列不得有空值，否则无法对应某个具体的学生。如果存在空值，则该表不完整，对应关系不符合实体完整性规则的约束条件。在物理

数据库中,表的主键强制执行实体完整性。

(2) 域完整性确保属性中只允许一个有效数据。域是属性可能值的范围,如整数、日期或字符。是否可以是空值也是域完整性的一部分。在物理数据库中,可以利用表中的数据类型和行可控性强制执行域完整性。

(3) 参照完整性。参照完整性是定义建立关系之间联系的主关键字与外部关键字引用的约束条件。关系数据库中通常包含多个存在相互联系的关系,关系与关系之间的联系是通过公共属性来实现的。

例如,在 teaching 数据库中,将 score 关系作为参照关系,将 student 关系作为被参照关系,以 studentno 作为两个关系进行关联的属性,则 studentno 是 student 关系的主键,是 score 关系的外键。score 关系通过外键 studentno 参照 student 关系。其中,公共属性 studentno 是一个关系 student(称为被参照关系)的主键,同时又是 score 关系(称为参照关系)的外键。

(4) 事务完整性。事务可以确保每个逻辑单元的工作(如插入 100 行或更新 1000 行数据)作为单个事务执行。事务可通过其 4 个基本属性检测数据库产品的质量,即原子性(全部执行或全部不执行)、一致性(数据库必须在一致的状态下开始及结束事务)、隔离性(一项事务不应该影响其他事务)和持久性(一旦提交,始终提交)。

(5) 用户定义完整性。对于数据完整性,除了前面 4 个普遍接受的定义,还添加了用户定义数据完整性。用户定义完整性则是根据应用环境的要求和实际的需要,对某一具体应用所涉及的数据提出约束性条件。这一约束机制一般不应由应用程序提供,而应由关系模型提供定义并检验,用户定义完整性主要包括字段有效性约束和记录有效性。

1.4 MySQL 数据库软件的使用

1.4.1 MySQL 5.7 的安装和配置步骤

在 Windows 操作系统下,MySQL 数据库的安装一般选择图形化界面安装,图形化界面包有完整的安装向导,安装和配置非常方便。

1. 安装准备

在安装之前,需要到 MySQL 数据库的官方网站(http://dev.mysql.com/downloads)上找到要安装的数据库版本并进行下载。当然,读者也可以直接在一些搜索引擎中搜索下载链接。在此介绍一下安装的简单过程,详细手工配置过程请参考本教材的辅导书《MySQL 数据库应用与开发习题解答与上机指导》第 16.3 节的内容。

MySQL 数据库的安装

2. 安装 MySQL 数据库的简单过程

MySQL 下载完成后,简单安装步骤如下。

(1) 双击 MySQL 安装程序(mysql-installer-community-5.7.17.0.msi),弹出如图 1-8 所示的“打开文件-安全警告”界面。

(2) 在图 1-8 所示的界面中单击“运行”按钮,进入 MySQL Installer 的协议许可界面,如图 1-9 所示,选中 I accept the license terms 复选框,表示接受用户安装时的许可协议。



图 1-8　安装 MySQL 时弹出的界面

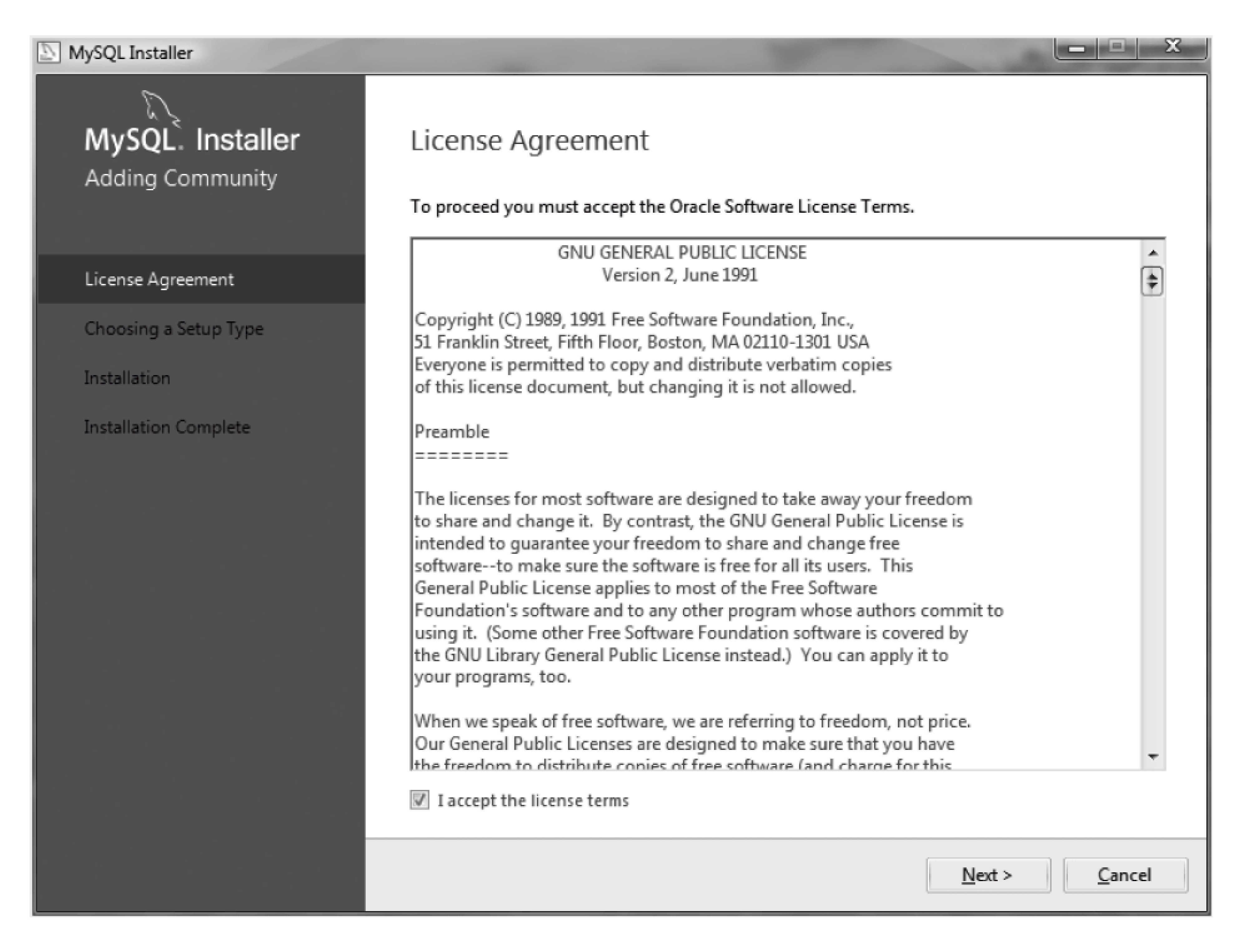


图 1-9　用户许可协议界面

(3) 单击 Next 按钮，进入安装类型选择界面，如图 1-10 所示。可以选择需要的版本，例如 Full(完整版)。左侧提供了 5 种安装类型，默认选中 Developer Default 选项。另外 4 项：Server only 表示仅作为服务器，Client only 表示仅作为客户端，Full 表示完全安装类型，Custom 表示自定义安装类型。该图右侧的 Installation Path 表示应用程序安装的路径，DataPath 表示数据库中数据文件的路径，默认情况下安装在 C 盘，用户可以根据实际情况选择其他磁盘。

(4) 单击 Next 按钮，进入将要安装或更新的应用程序界面，如图 1-11 所示。

(5) 单击 Execute 按钮，进入账户和角色界面，按要求设置 Root 账户密码，如图 1-12 所示。

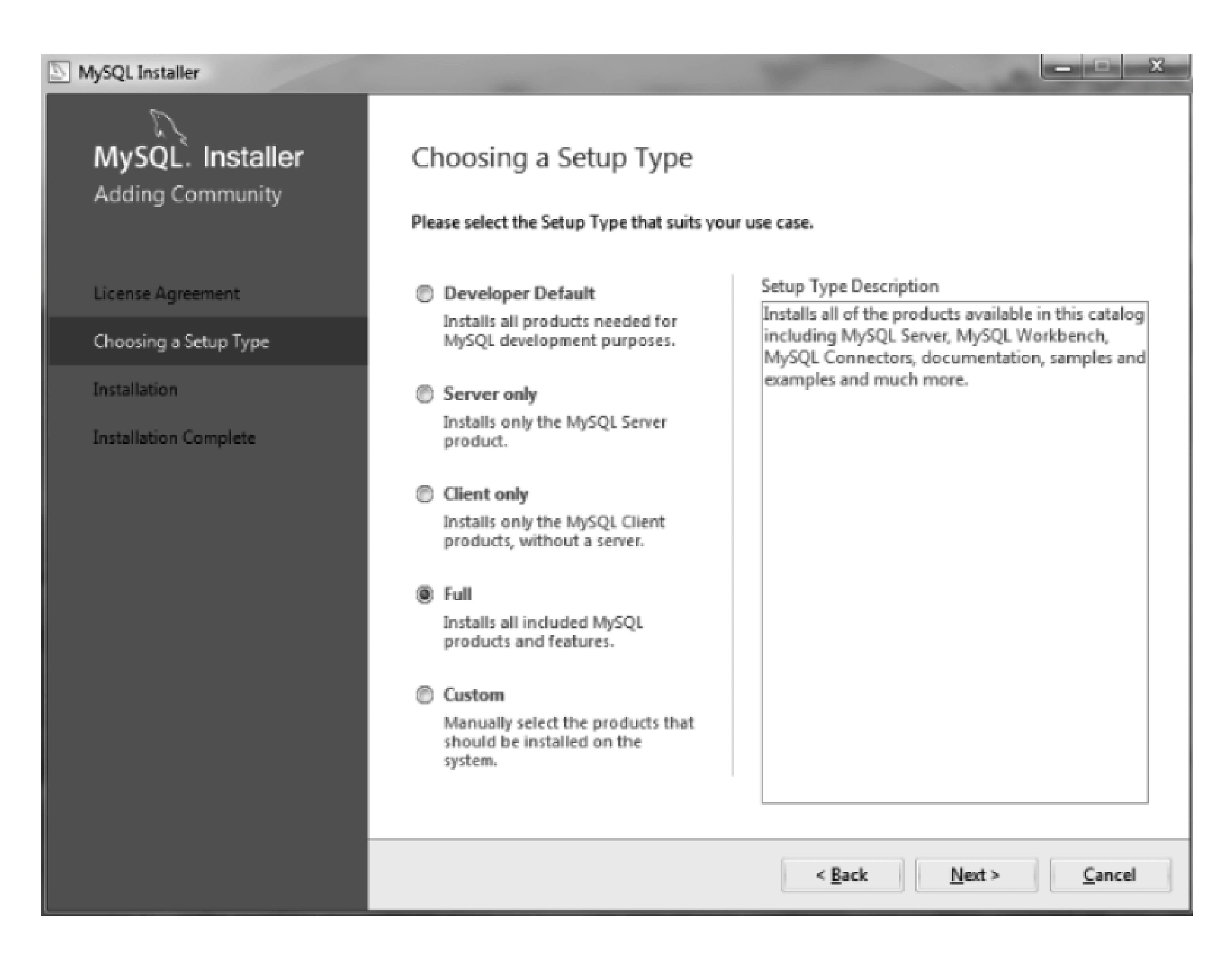


图 1-10　安装类型选择界面

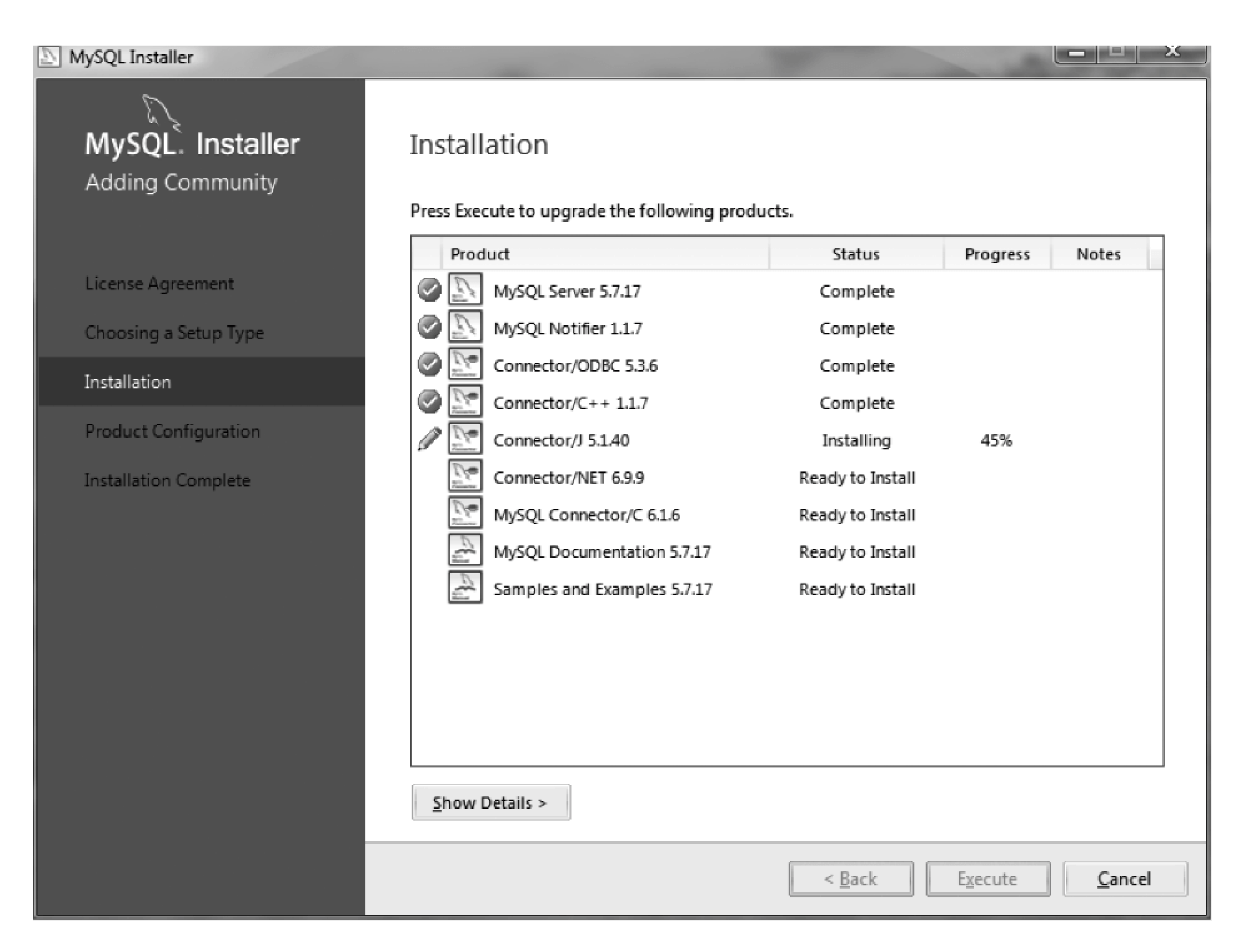


图 1-11　将要安装或更新的应用程序

（6）单击 Next 按钮，如图 1-13 所示，进入开始安装和配置 MySQL 服务器界面，依次按照提示，保持默认选择，就可以进入安装完成后的界面，如图 1-14 所示。单击图 1-14 所示中的 Finish 按钮就可以运行 MySQL 数据库了。

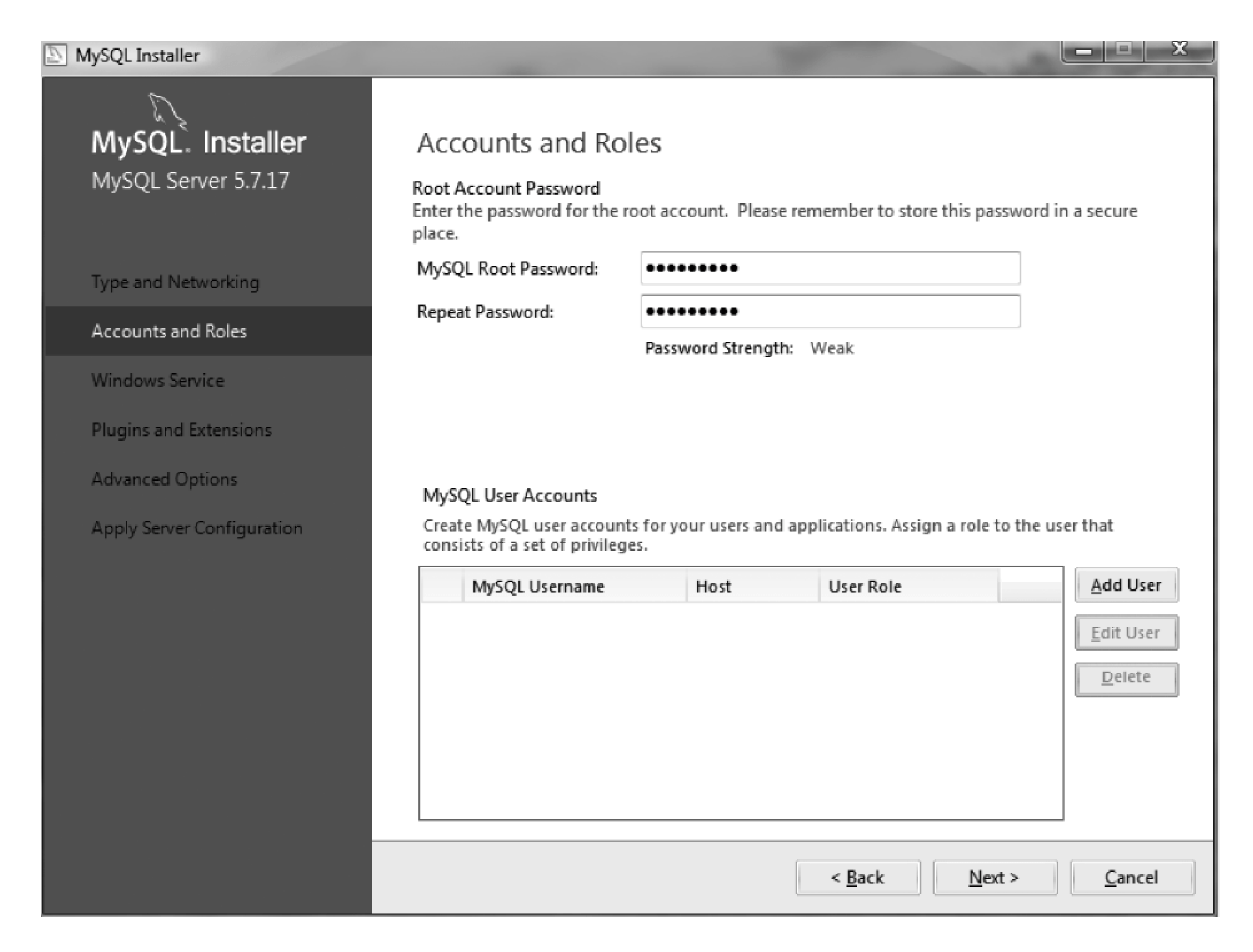


图 1-12　为 Root 用户设置密码

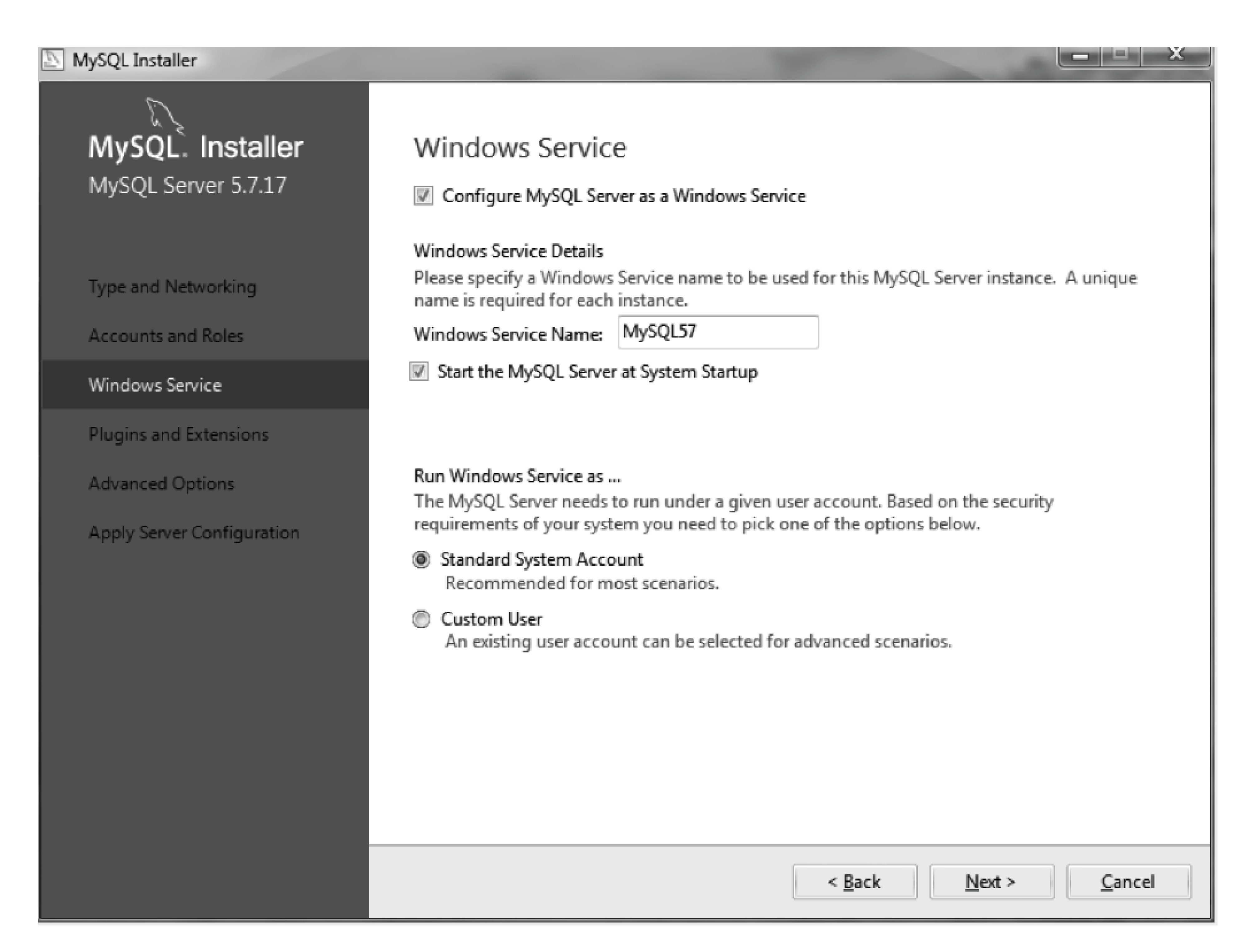


图 1-13　开始安装服务器

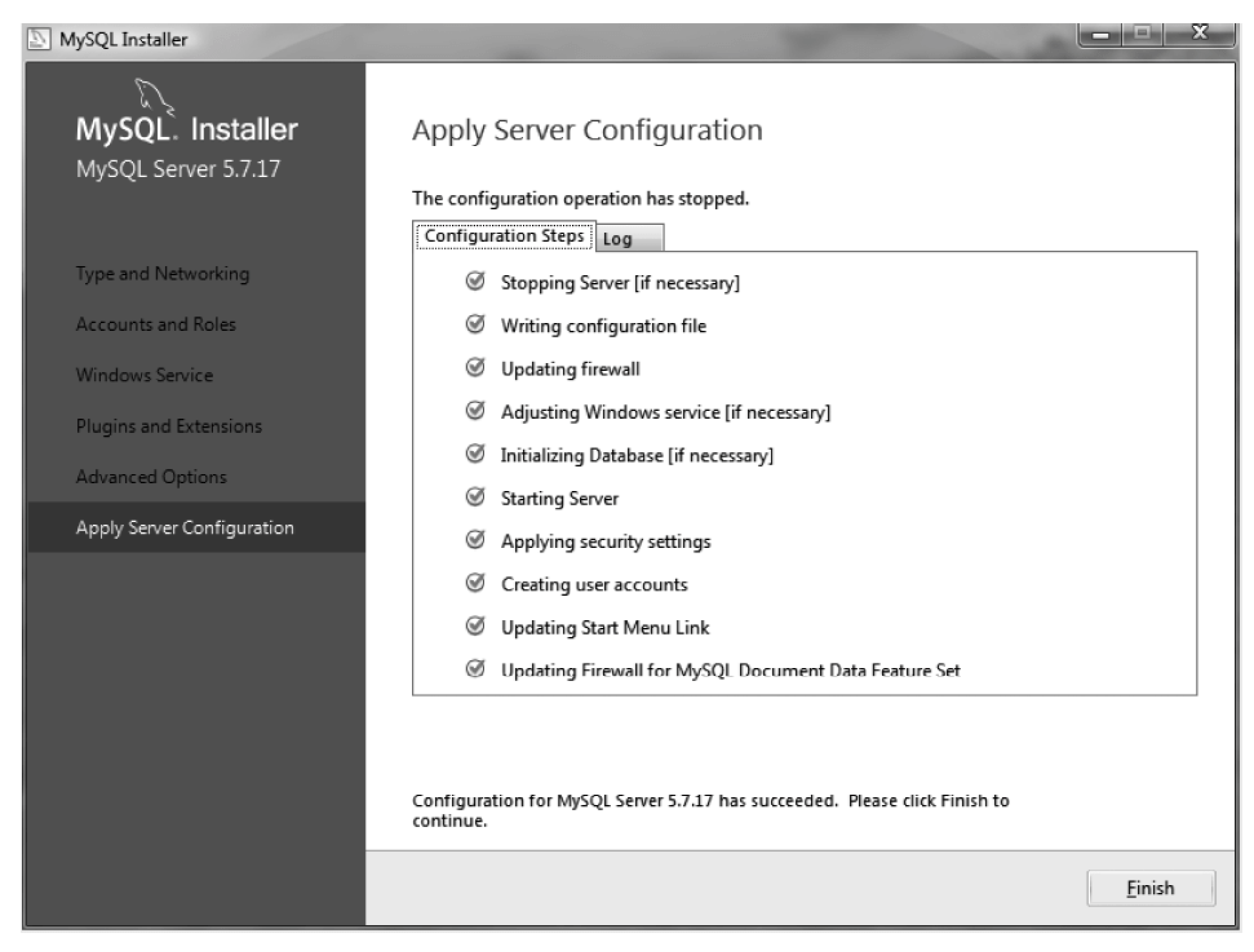


图 1-14　安装完成

(7) 在系统的“开始”菜单中可以找到与 MySQL 数据库有关的文件。

3. MySQL 服务的配置



MySQL 服务的配置

下面介绍环境变量的设置方法。其步骤如下：

(1) 右击“计算机”图标，在弹出的快捷菜单中选择“属性”命令，在弹出的“控制面板主页”中单击“高级系统设置”标签，弹出“系统属性”对话框，如图 1-15 所示。



图 1-15　“系统属性”对话框

(2) 在“系统属性”对话框中，打开“高级”选项卡，单击“环境变量”按钮，弹出“环境变量”对话框，如图 1-16 所示。



图 1-16 “环境变量”对话框

(3) 在“环境变量”对话框中，定位到“系统变量”中的 path 选项，单击“编辑”按钮，将弹出“编辑系统变量”对话框，如图 1-17 所示。

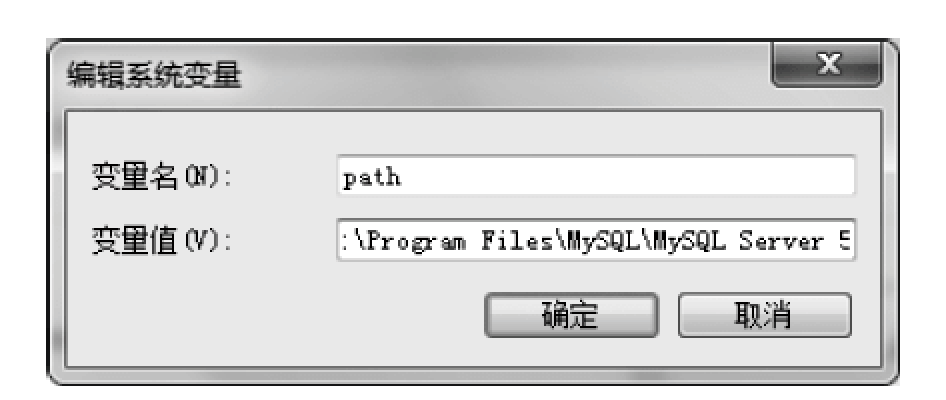


图 1-17 “编辑系统变量”对话框

(4) 在“编辑系统变量”对话框中，将 MySQL 服务器的 bin 文件夹位置(C:\Program Files\MySQL\MySQL Server 5.7\bin)添加到变量值文本框中，注意要使用“;”与其他变量值进行分隔，最后，单击“确定”按钮。

(5) 环境变量设置完成后，再使用 MySQL 命令即可成功连接 MySQL 服务器。

(6) 断开 MySQL 服务器。连接到 MySQL 服务器后，可以通过在 MySQL 提示符下输入 exit 或者 quit 命令断开 MySQL 连接，格式如下：

```
mysql> quit;
```

1.4.2 MySQL 的工作流程

MySQL 数据库的工作流程如图 1-18 所示，具体包括如下内容：

(1) 操作系统用户启动 MySQL 服务。

(2) MySQL 服务启动期间，首先将配置文件中的参数信息读入服务器内存。

(3) 根据 MySQL 配置文件的参数信息或者编译 MySQL 时参数的默认值生成一个服

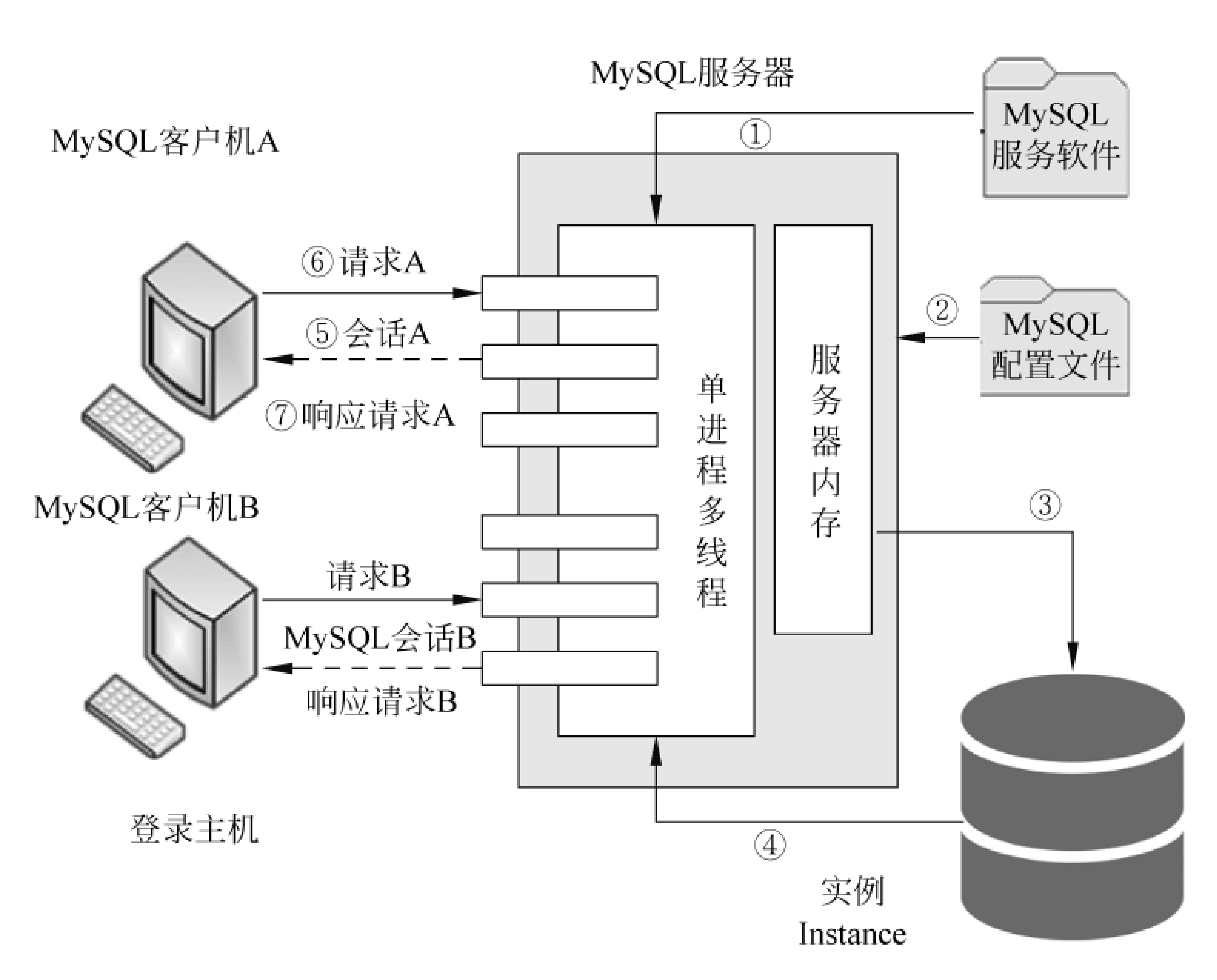


图 1-18　MySQL 数据库的工作流程

务实例进程 Instance。

(4) MySQL 服务实例进程派生出多个线程为多个客户机提供服务。

(5) 数据库用户访问 MySQL 服务器的数据时,首先需要选择一台登录主机,然后在该登录主机上开启客户机,输入正确的账户名、密码,建立一条客户机与服务器之间的"通信链路"。

(6) 接着数据库用户就可以在 MySQL 客户机上输入 MySQL 命令或 SQL 语句,这些 MySQL 命令或 SQL 语句沿着该通信链路传送给 MySQL 服务实例,这个过程称为客户机向 MySQL 服务器发送请求。

(7) MySQL 服务实例负责解析这些 MySQL 命令或 SQL 语句,并选择一种执行计划运行这些 MySQL 命令或 SQL 语句,然后将执行结果沿着通信链路返回给客户机,这个过程称为 MySQL 服务器向 MySQL 客户机返回响应。

(8) 数据库用户关闭 MySQL 客户机,通信链路被断开,该客户机对应的 MySQL 会话结束。

1.4.3　MySQL 数据库工具简介

MySQL 数据库管理系统提供了许多命令行工具,这些工具可以用来管理 MySQL 服务器、对数据库进行访问控制、管理 MySQL 用户以及数据库备份和恢复工具等。另外,MySQL 还提供了图形化的管理工具,这使得对数据库的操作更加简单。

1. MySQL 服务器端的常用工具

MySQL 服务器端实用工具程序如下。

(1) mysqld：SQL 后台程序(即 MySQL 服务器进程)。该程序必须运行之后,客户端才能通过连接服务器来访问数据库。

（2）mysqld_safe：服务器启动脚本。在 UNIX 和 NetWare 中推荐使用 mysqld_safe 来启动 mysqld 服务器。mysqld_safe 增加了一些安全特性，例如当出现错误时重启服务器并向错误日志文件写入运行时间信息。

（3）mysql.server：服务器启动脚本。在 UNIX 中的 MySQL 分发版包括 mysql.server 脚本。该脚本用于使用包含为特定级别的、运行启动服务的脚本的、运行目录的系统。它调用 mysqld_safe 来启动 MySQL 服务器。

（4）mysql_multi：服务器启动脚本，可以启动或停止系统上安装的多个服务器。

（5）myisamchk：用来描述、检查、优化和维护 MyISAM 表的实用工具。

（6）mysqlbug：MySQL 缺陷报告脚本。它可以用来向 MySQL 邮件系统发送缺陷报告。

（7）mysql_install_db：该脚本用默认权限创建 MySQL 授权表。通常只是在系统上首次安装 MySQL 时执行一次。

2. MySQL 客户端常用工具

MySQL 客户端实用工具程序如下：

（1）myisampack：压缩 MyISAM 表以产生更小的只读表的一个工具。

（2）mysql：交互式输入 SQL 语句或从文件以批处理模式执行它们的命令行工具。

（3）mysqlaccess：检查访问主机名、用户名和数据库组合的权限的脚本。

（4）MySQLadmin：执行管理操作的客户程序，例如创建或删除数据库，重载授权表，将表刷新到硬盘上，以及重新打开日志文件。MySQLadmin 还可以用来检索版本、进程，以及服务器的状态信息。

（5）mysqlbinlog：从二进制日志读取语句的工具。在二进制日志文件中包含执行过的语句，可用来帮助系统从崩溃中恢复。

（6）mysqlcheck：检查、修复、分析以及优化表的表维护客户程序。

（7）mysqldump：将 MySQL 数据库转存到一个文件（例如 SQL 语句或 Tab 分隔符文本文件）的客户程序。

（8）mysqlhotcopy：当服务器在运行时，快速备份 MyISAM 或 ISAM 表的工具。

（9）mysql import：使用 load data infile 将文本文件导入相关表的客户程序。

（10）mysqlshow：显示数据库、表、列以及索引相关信息的客户程序。

（11）perror：显示系统或 MySQL 错误代码含义的工具。

1.4.4 MySQL 的启动和登录

MySQL 数据库安装完成后可以在 DOS 窗口登录数据库执行语句。MySQL 数据库分为客户端和服务器端，下面将介绍启动 MySQL 服务和登录 MySQL 数据库两部分内容。

1. 启动 MySQL 服务

在安装 MySQL 数据库的过程中，可以设置 MySQL 服务的自动启动。如果 MySQL 服务没有启动，Windows 操作系统通常通过两种方式进行启动。


启动 MySQL 服务

（1）CMD 控制台启动。这种方式非常简单，net start mysql57 表示启动 MySQL 服务，net stop mysql57 表示关闭 MySQL 服务。

(2) 手动启动。执行“开始”→“设置”→“控制面板”→“管理工具”→“服务”命令进行设置，打开的窗口如图 1-19 所示(执行“开始”→“运行”命令并输入 services. msc 后按 Enter 键也可以弹出图 1-19 所示的窗口)。



图 1-19 “服务”窗口

从图 1-19 中可以看到 MySQL 服务已经启动，而且服务的启动类型是自动启动。MySQL 服务启动后，可以在 Windows 的任务管理器中查看服务是否已经运行，此时，可以通过客户端来访问 MySQL 数据库。

另外，在图 1-20 中可以更改 MySQL 服务的启动类型，选中 MySQL 57 服务项右击，在弹出的快捷菜单中选择“属性”命令，弹出如图 1-20 所示的对话框。

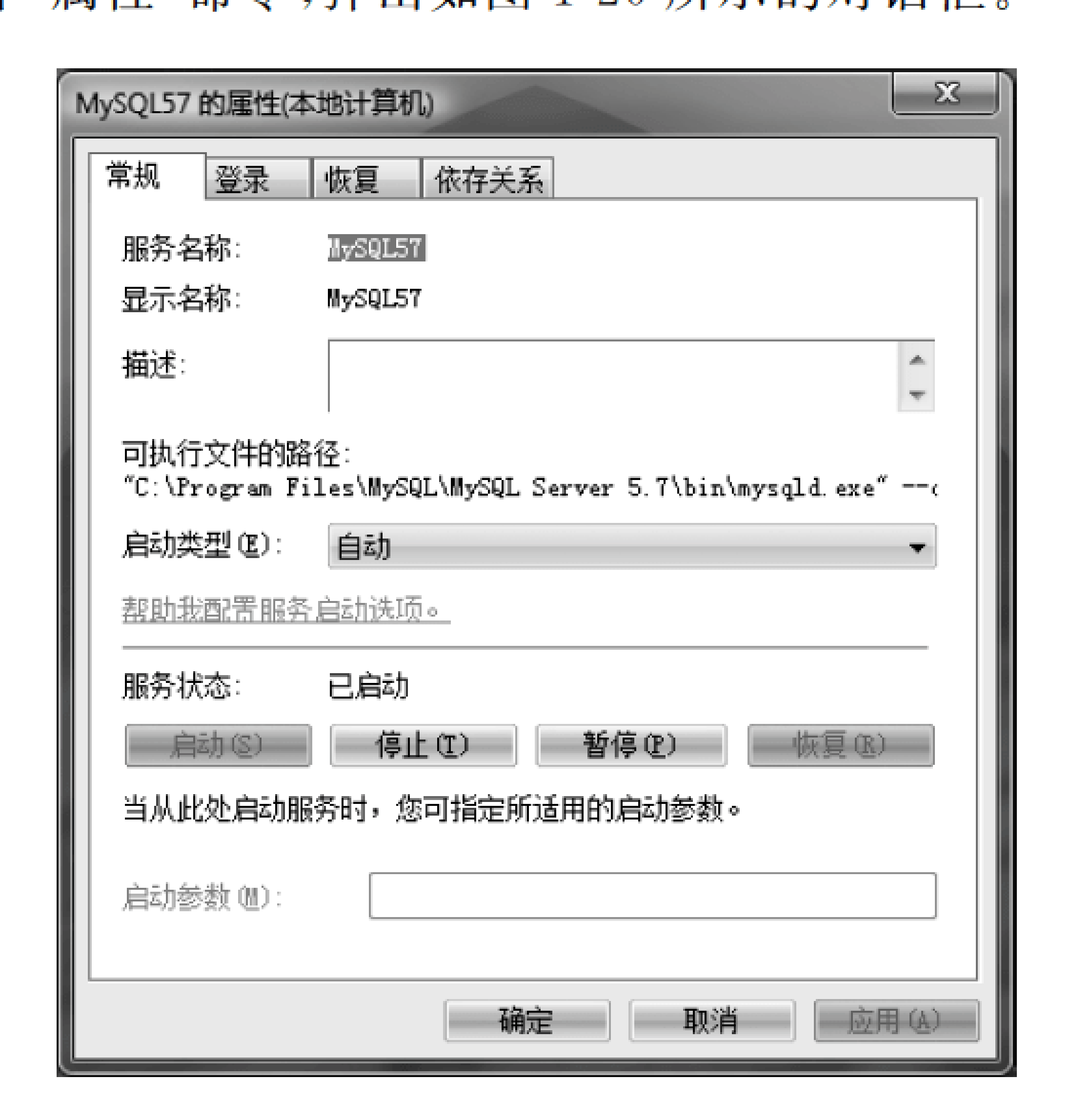


图 1-20 更改服务的启动类型

从图 1-20 中可以看到，可以更改服务状态为“停止”“暂停”和“恢复”，还可以设置服务的启动类型。在“启动类型”下拉列表框中可以选择“自动”“手动”和“已禁用”选项，说明如下。

- 自动：MySQL 服务自动启动，可以手动将服务状态变为停止、暂停和重新启动等。如果读者经常练习 MySQL 数据库的操作，最好将 MySQL 设置为自动启动，这样可以避免每次手动启动 MySQL 服务。
- 手动：MySQL 服务需要手动启动，启动后可以改变服务状态，如停止和暂停等。如果读者使用 MySQL 数据库的频率很低，可以考虑将 MySQL 服务设置为手动启动，这样可以避免 MySQL 服务长时间占用系统资源。
- 已禁用：MySQL 服务不能启动，也不能改变服务状态。

2. 登录 MySQL 数据库



MySQL 的登录和退出

MySQL 服务启动后，可以通过客户端来登录 MySQL 数据库。Windows 操作系统下有两种登录 MySQL 数据库的方式：一种是执行 CMD 命令，在打开的 DOS 窗口中以命令行的方式登录 MySQL 数据库；另一种是在 MySQL 客户端直接登录数据库。

（1）在 DOS 窗口中登录 MySQL 数据库。通过 DOS 窗口登录 MySQL 数据库执行语句时，可以执行“开始”→“运行”命令，在弹出的对话框中输入“cmd”后按 Enter 键，即可进入 DOS 窗口。

在 DOS 窗口中首先进入当前 MySQL 的安装目录下，然后再执行 MySQL 语句登录数据库，登录成功后会出现“Welcome to the MySQL monitor”的欢迎语。结果如下：

```
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。
C:\Users\Administrator>net start mysql57
请求的服务已经启动。
请键入 NET HELPMSG 2182 以获得更多的帮助。
C:\Users\Administrator>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 5.7.17-log MySQL Community Server (GPL)
Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved. Oracle is a
registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks
of their respective owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>
```

上述执行语句的代码中，-u 后面紧跟着数据库的用户名，此处使用 root 用户进行登录；-p 表示用户密码，按 Enter 键输入密码，输入的密码使用星号(*)表示。

（2）MySQL 客户端登录数据库。在 Windows 7 操作系统中，执行“开始”→“所有程序”→MySQL→MySQL 5.7 目录，该目录下包含 MySQL Command Line Client 和 MySQL Command Line Client-Unicode 两个选项。它们都是 MySQL 客户端的命令行工具，也可以称为 MySQL 的 DOS 窗口或控制台。通过在控制台中执行语句可以登录 MySQL 数据库，然后执行其他的相关 SQL 语句进行操作。

打开 MySQL Command Line Client 弹出 MySQL 客户端的控制台，直接输入密码后按 Enter 键，登录成功后的输出内容与上述代码一致，这里不再显示。

无论是 DOS 窗口还是控制台，登录成功后除了显示欢迎语外，还包含一些说明性的语

句，这些语句的说明如下：

- Commands end with ；or \g：说明 MySQL 控制台下的命令是以分号(；)或"\g"来结束的，遇到这个结束符就开始执行命令。
- Your MySQL connection id is 5：id 表示 MySQL 数据库的连接次数，如果数据库是新安装的，且是第一次登录，则显示 1。如果安装成功后已经登录过，将会显示其他的数字。
- Server version：Server version 之后的内容表示当前数据库的版本，这里安装的版本是 5.7.17-enterprise-commercial-advanced。
- Type 'help；' or '\h' for help：表示输入"help；"或者"\h"可以看到帮助信息。
- Type '\c' to clear the current input statement：表示遇到\c 就清除当前输入的语句。

MySQL 数据库安装完成后，可能会根据实际情况更改 MySQL 数据库的某些配置。一般可以通过两种方式进行更改：一种是通过配置向导进行更改；另一种是通过手动方式更改 MySQL 数据库的某些配置。手动更改方式虽然比较困难，但是这种配置方式更加灵活。

在控制台中执行语句时，如果执行的语句不合法或者错误，则会输出有关的错误信息。例如，在命令中执行"select year(current_data)；"语句时出现 1054 错误。详细错误代码介绍请参考本教材的辅导书《MYSQL 数据库应用与开发习题解答与上机指导》第 16.4 节的内容。

登录 MySQL 数据库后就可以执行一些语句查看操作结果了，例如，查看系统的帮助信息、系统当前时间和当前版本等。

例如，登录 MySQL 数据库成功后可以直接输入"help；"或"\h"查看帮助信息，直接在控制台中输入"help；"语句按 Enter 键，输出结果如下：

```
mysql > help
List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?             (\?) Synonym for 'help'.
clear         (\c) Clear the current input statement.
connect       (\r) Reconnect to the server. Optional arguments are db and host.
delimiter     (\d) Set statement delimiter.
ego           (\G) Send command to mysql server, display result vertically.
exit          (\q) Exit mysql. Same as quit.
go            (\g) Send command to mysql server.
help          (\h) Display this help.
note          (\t) Don't write into outfile.
print         (\p) Print current command.
prompt        (\R) Change your mysql prompt.
quit          (\q) Quit mysql.
rehash        (\#) Rebuild completion hash.
source        (\.) Execute an SQL script file. Takes a file name as an argument.
status        (\s) Get status information from the server.
tee           (\T) Set outfile [to_outfile]. Append everything into given outfile.
use           (\u) Use another database. Takes database name as argument.
charset       (\C) Switch to another charset. Might be needed for processing binlog
with multi - byte charsets.
warnings      (\W) Show warnings after every statement.
```

```
nowarning     (\w) Don't show warnings after every statement.
resetconnection(\x) Clean session context.
For server side help, type 'help contents'
mysql >
```

当然也可以利用 select 语句查询一些系统参数。select 语句可以用于查询，MySQL 数据库中经常会使用到该语句。它类似于其他编程语言中的 print 或者 write，可以使用它来显示一个字符串、数字或数学表达式的结果等。

例如，如果要查看系统的当前时间，可以直接在控制台中输入并执行“select now()；”语句即可，执行结果如下：

```
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation.保留所有权利。
C:\Users\Administrator > mysql  - u root  - p
Enter password: ******
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.17 - log MySQL Community Server (GPL)
Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql > select now();
+---------------------+
| now()               |
+---------------------+
| 2017 - 05 - 03 10:10:00 |
+---------------------+
1 row in set (0.05 sec)
mysql >
```

1.4.5 MySQL 的图形管理工具

MySQL 的图形管理工具有很多，常用的有 MySQL Workbench、phpMyAdmin 和 Navicat 等软件。本书选用 MySQL Workbench 软件作为可视化操作的管理工具。详细的 phpMyAdmin 软件和 Navicat 软件的安装和使用介绍请参看本教材的辅导书《MYSQL 数据库应用与开发习题解答与上机指导》第 17 章的内容。

MySQL Workbench 为数据库管理员、程序开发者和系统规划师提供可视化设计、模型建立以及数据库管理功能。它包含了用于创建复杂的数据建模 E-R 模型，正向和逆向数据库工程，也可以用于执行通常需要花费大量时间和难以变更和管理的文档任务。



Workbench 的安装和连接

下面以安装 mysql-workbench-community-6. 2. 5-win32 版本的 MySQL Workbench 工具为例，简单介绍 MySQL Workbench 工具的安装过程。MySQL Workbench 软件的安装步骤如下。

(1) 双击安装文件“mysql-workbench-community-6. 2. 5-win32. msi”。进入“安全警告”界面，单击 Next 按钮，进入安装向导界面，如图 1-21 所示。

(2) 单击 Next 按钮，进入选择安装文件夹界面，单击 Change 按钮，可以选择合适的路径，如图 1-22 所示。



图 1-21　开始安装

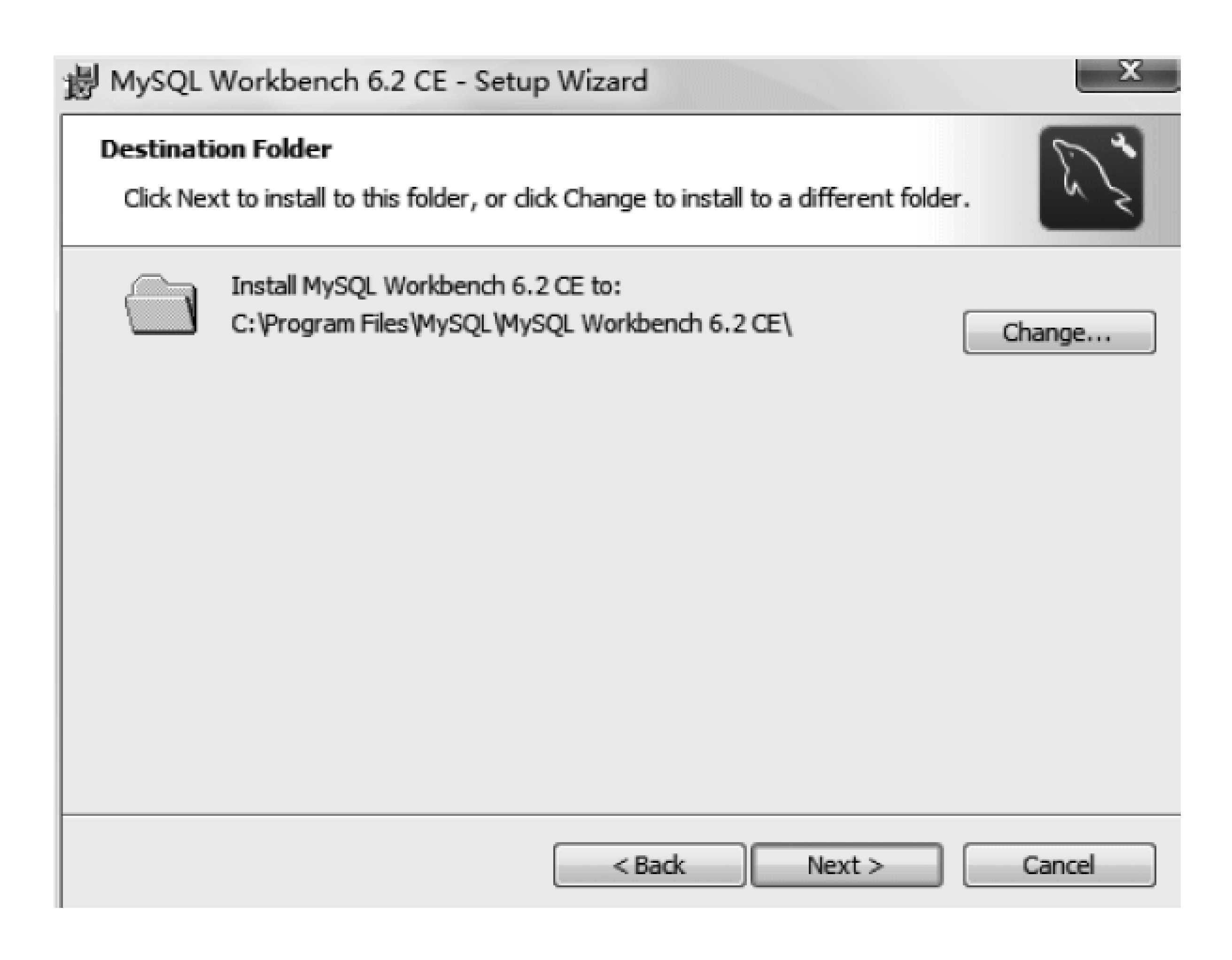


图 1-22　设置安装路径

(3) 依次单击 Next 按钮，依次进入选择安装类型和准备安装项目界面，如图 1-23 所示。

(4) 单击 Install 按钮，进入安装过程，最后单击 Finish 按钮，即可完成 MySQL Workbench 软件的安装，如图 1-24 所示。

(5) 选择“开始”→“所有程序”命令，按照如图 1-25 所示，单击 MySQL 下的 MySQL Workbench 6.2 CE 命令，即可进入如图 1-26 所示的 MySQL Workbench 界面，接下来就可以利用 MySQL Workbench 软件实现 MySQL 数据库的可视化操作了。

在图 1-26 中，MySQL Workbench 工具包含以下 4 个基本功能区域。

主菜单：实现 MySQL 的主要功能操作。

Shortcut(快捷方式)：完整的可视化数据库设计和建模。

MySQL Connections：连接信息。

Models：连接方式、MySQL Workbench 工具版本信息。

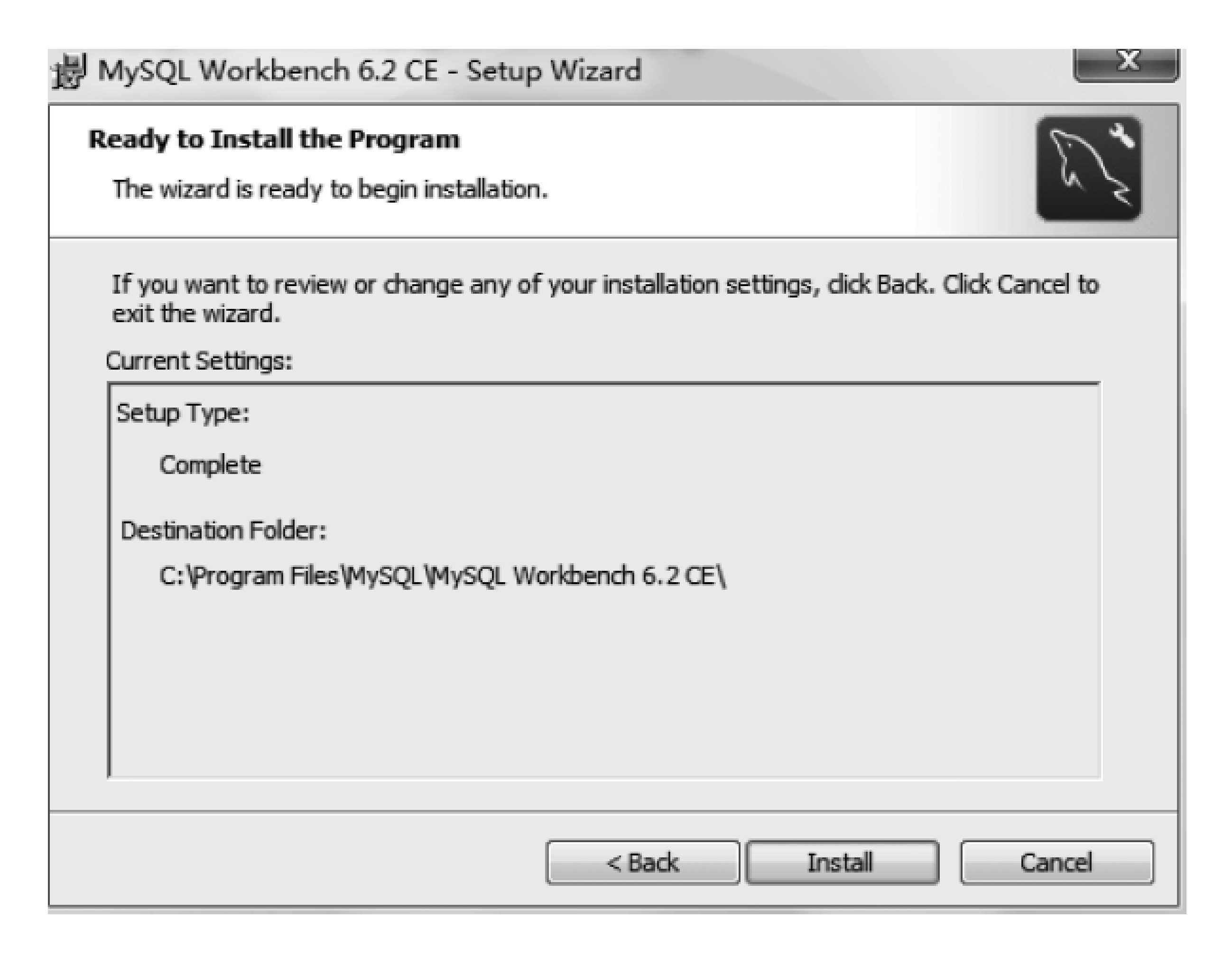


图 1-23　安装准备



图 1-24　完成安装



图 1-25　执行 MySQL Workbench 6.2 CE 命令



图 1-26　MySQL Workbench 工具页面

（6）创建连接。在图形界面中最常用的还是对数据库的基本操作，例如，执行 SQL 语句实现数据库的添加、数据库表的添加和数据添加、删除以及修改等操作。如果要实现这些操作首先要连接到数据库，单击主菜单 Database→Manage Connections 命令，弹出 Manage Server Connections 对话框，在如图 1-27 所示的对话框中输入连接名称，输入完成后单击 Test Connection 按钮进行测试，输入 Root 密码，测试成功后如图 1-28 所示，单击 OK 按钮。返回主界面单击 Close 按钮即可完成连接。

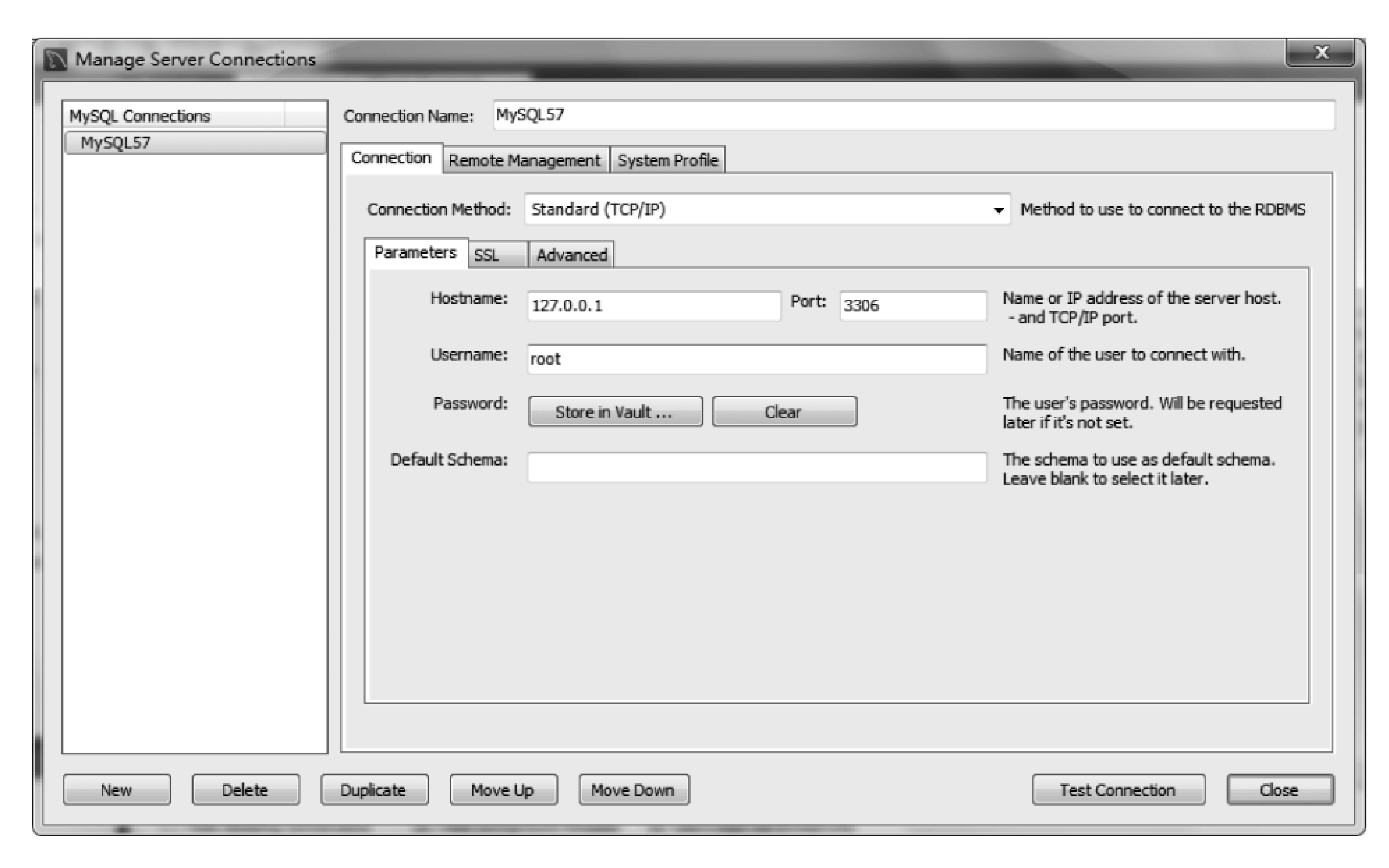


图 1-27　连接参数设置

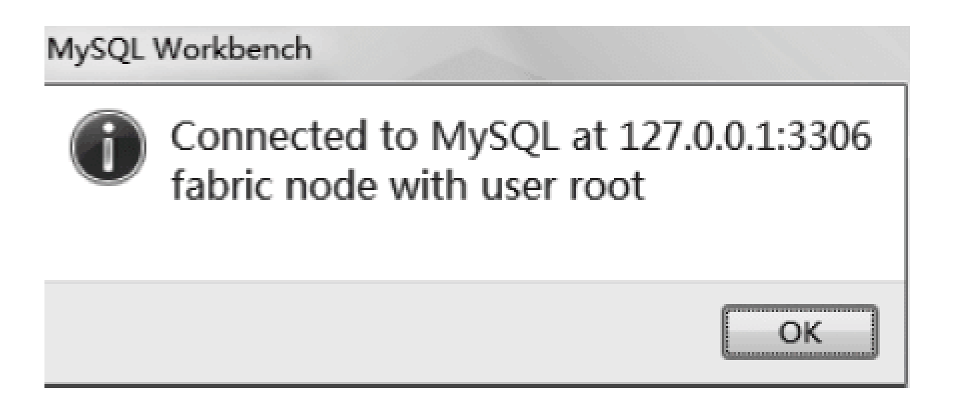


图 1-28　连接成功

1.5　小　　结

本章介绍了数据的基本概念、数据模型、数据库分类以及 MySQL 数据库的基本知识。还介绍了有关数据库设计的基本方法，为后续章节的学习打下基础。关于数据库范式的知识难度比较大，读者只要能够了解相关知识就行了。学习本章需要重点掌握如下内容：

- 数据库管理系统的功能和组成。
- 关系数据库的基本理论。
- 数据库系统的基本组成。
- 安装、启动和配置 MySQL 的基本过程。

习　题　1

1. 选择题

(1) 数据模型的 3 要素不包括________。

A. 数据结构　　B. 数据操作　　C. 数据类型　　D. 完整性约束

(2) 关系运算不包括________。

A. 连接　　B. 投影　　C. 选择　　D. 查询

(3) 表 1-1 所示的学生信息表中的主键为________。

A. studentno　　B. sex　　C. birthdate　　D. sname

(4) 下面的数据库产品中，________是开源数据库。

A. Oracle　　B. SQL Server　　C. MySQL　　D. DB2

(5) E-R 概念模型中，信息的 3 种概念单元不包括________。

A. 实体型　　B. 实体值　　C. 实体属性　　D. 实体间联系

(6) Linux 和 UNIX 操作平台上不能使用________作为数据库。

A. Oracle　　B. DB2　　C. MySQL　　D. SQL Server

(7) E-R 图是数据库设计的工具之一，一般适用于建立数据库的________。

A. 概念模型　　B. 结构模型　　C. 物理模型　　D. 逻辑模型

(8) SQL 语言又称________。

A. 结构化定义语言　　B. 结构化控制语言

C. 结构化查询语言　　D. 结构化操纵语言

(9) 从 E-R 模型向关系模型转换，一个 M∶N 的联系转换成一个关系模式时，该关系

模式的键是________。

A. M 端实体的键
B. N 端实体的键
C. M 端实体键与 N 端实体键的组合
D. 重新选取其他属性

(10) DB、DBS 和 DBMS 3 者之间的关系是________。

A. DB 包括 DBMS 和 DBS
B. DBS 包括 DB 和 DBMS
C. DBMS 包括 DB 和 DBS
D. 不能相互包括

2. 简答题

(1) 什么是数据库管理系统? 举出日常生活中一些应用数据库的实际范例。

(2) 说明 MySQL 数据库管理系统基本系统架构拥有哪 4 大模块。

(3) 举例说明 3 种关系运算的特点。

3. 上机练习

(1) 从 MySQL 的官方网站(https://dev.mysql.com/downloads/)下载 MySQL 5.7.19 版本的数据库,解压缩后进行安装,观察安装成功后的效果。

(2) 在控制台中完成操作。在启动 MySQL 服务后打开控制台,在控制台中输入密码完成 MySQL 的登录,并且执行 select 命令的相关语句查看系统的当前日期。

(3) 说明在"服务"窗口中,MySQL 服务器的启动和关闭方法。

第2章 MySQL语言基础

MySQL语言是一系列操作数据库及数据库对象的命令语句，因此使用MySQL数据库就必须掌握构成其基本语法和流程语句的语法要素，这主要包括常量、变量、关键词、运算符、函数、表达式和控制流语句等。而字符集是最基本的MySQL脚本组成部分，也是MySQL数据库对象的描述符号。

本章主要介绍MySQL的基本语法要素。

2.1 MySQL的基本语法要素

MySQL能够支持39种字符集和127个校对原则。本节着重介绍latin1、UTF-8和GB 2312字符集的用法，并学习掌握修改默认字符集的方法及在实际应用中如何选择合适的字符集，避免在向数据表中录入中文数据、查询包括中文字符的数据时，会出现类似"?"这样的乱码现象。同时也介绍常量、变量、标识符和关键词的使用。

2.1.1 字符集与标识符

1. 字符集及字符序概念

字符(Character)是指人类语言中最小的表义符号。例如'A''7'"%"等字母、数字和特殊符号。字符校对原则(Collation)也称为字符序，是指在同一字符集内字符之间的比较规则。字符集只有在确定字符序后，才能在一个字符集上定义什么是等价的字符，以及字符之间的大小关系。每个字符序唯一对应一种字符集，但一个字符集可以对应多种字符校对原则，其中有一个是默认字符校对原则(Default Collation)。

MySQL服务器默认的字符集是latin1。MySQL的字符集支持可以细化到4个层次：服务器(Server)、数据库(DataBase)、数据表(Table)和连接层(Connection)。

MySQL中的字符序名称遵从命名惯例：以字符序对应的字符集名称开头；以_ci(表示大小写不敏感)、_cs(表示大小写敏感)或_bin(表示按编码值比较)结尾。例如，在字符序"utf8_general_ci"下，字符'a'和'A'是等价的。

如果不进行设置，那么连接层级、客户端级和结果返回级、数据库级、表级、字段级都默认使用latin1字符集。

2. 字符序与常用字符集

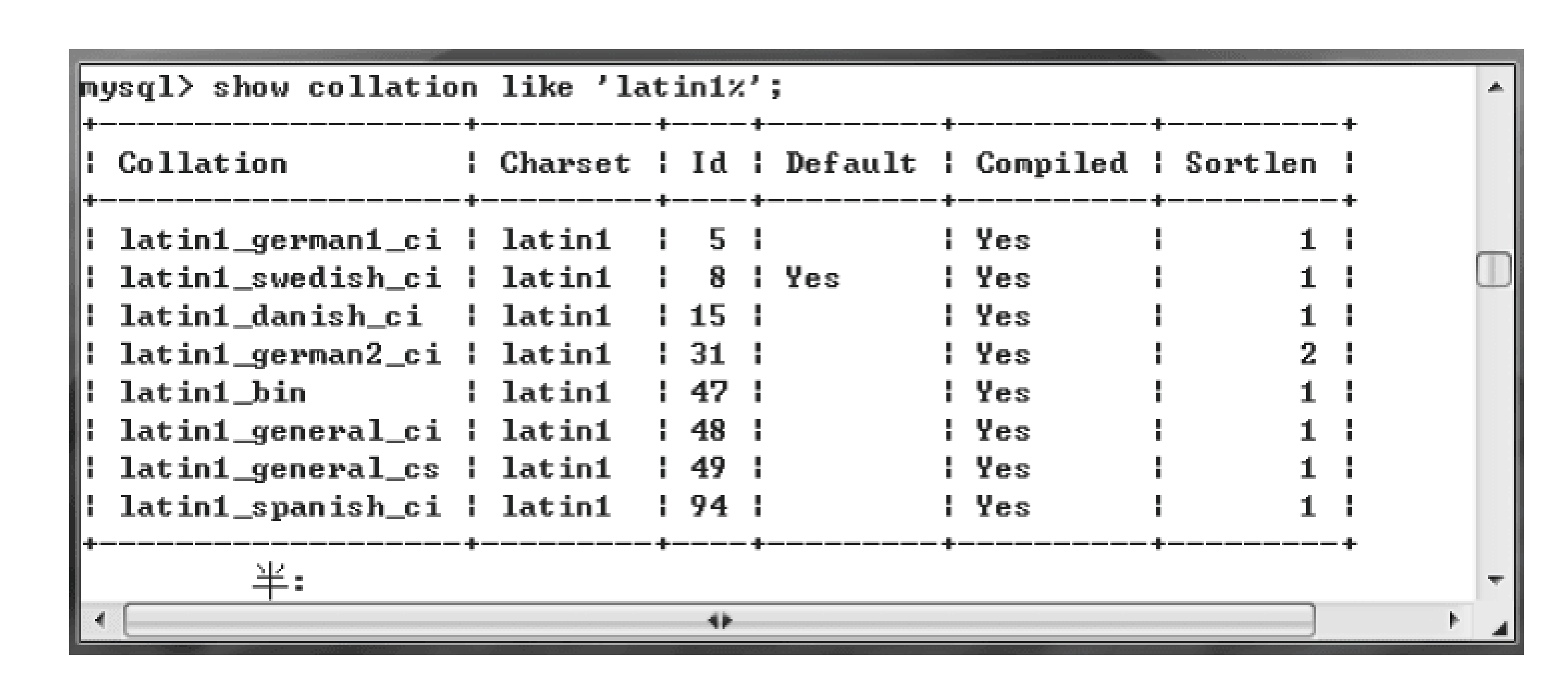
MySQL的字符集通过show character set语句查看。在命令窗口中执行如下命令，即可查看到MySQL的39种字符集。

```
mysql> show character set;
```

对于任何一个给定的字符集至少有一个校对原则，也可能有几个校对原则。例如，执行显示 latin1 系列的命令：

```
mysql> show collation like 'latin1%';
```

结果如图 2-1 所示。



mysql> show collation like 'latin1%';

Collation	Charset	Id	Default	Compiled	Sortlen
latin1_german1_ci	latin1	5		Yes	1
latin1_swedish_ci	latin1	8	Yes	Yes	1
latin1_danish_ci	latin1	15		Yes	1
latin1_german2_ci	latin1	31		Yes	2
latin1_bin	latin1	47		Yes	1
latin1_general_ci	latin1	48		Yes	1
latin1_general_cs	latin1	49		Yes	1
latin1_spanish_ci	latin1	94		Yes	1

半:

图 2-1　latin1 系列字符序

说明：

(1) 系统启动时默认的字符集是 latin1，latin1 是一个 8 位字符集，字符集名称为 ISO 8859-1Latin 1，也简称为 ISO Latin-1。latin1 把位于 128～255 之间的字符用于拉丁字母表中特殊语言字符的编码，也因此而得名。

(2) UTF-8(8-bit Unicode Transformation Format)被称为通用转换格式，是针对 Unicode 字符的一种变长字符编码。该字符集是用以解决国际上字符的一种多字节编码，它对英文使用 8 位(即 1 个字节)，中文使用 24 位(3 个字节)来编码。UTF-8 包含全世界所有国家需要用到的字符，是国际编码，通用性强。UTF-8 编码的文字可以在各国支持 UTF-8 字符集的浏览器上显示。例如，如果是 UTF-8 编码，则在外国人的英文 IE 上也能显示中文，他们无须下载 IE 的中文语言支持包。

(3) GB 2312 是简体中文字符集，GBK 是对 GB 2312 的扩展，其校对原则分别为 gb2312_chinese_ci、gbk_chinese_ci。GBK 是在国家标准 GB 2312 基础上扩容后兼容 GB 2312 的标准。GBK 的文字编码是用双字节来表示的，即不论中、英文字符均使用双字节来表示，为了区分中文，将其最高位都设定成 1。GBK 包含全部中文字符，是国家编码，通用性比 UTF-8 差，不过 UTF-8 占用的数据库比 GBK 大。

GBK、GB 2312 等与 UTF-8 之间都必须通过 Unicode 编码才能相互转换。对于一个网站、论坛来说，如果英文字符较多，则建议使用 UTF-8 节省空间。不过现在很多论坛的插件一般只支持 GBK。

3. 标识符和关键字

MySQL 的脚本由一条或多条 MySQL 语句组成，保存时脚本文件后缀名一般为.sql。在控制台下，MySQL 客户端也可以对语句进行单句的执行而不用保存为.sql 文件。而这些语句中最常用的就是标识符和关键字。

(1) 标识符。标识符用来命名一些对象，如数据库、表、列、变量等，以便在脚本中的其

他地方引用。MySQL 标识符命名规则稍微有点烦琐，其通用命名规则是：标识符由以字母或下画线开头的字母、数字或下画线(_)序列组成。

对于标识符是否区分大小写取决于当前的操作系统，Windows 下是不敏感的，但对于大多数 linux\UNIX 系统，这些标识符大小写是敏感的。

(2) 关键字。MySQL 的关键字众多，不同版本的 MySQL 语言关键字也略有变化。MySQL 5.7 大约有 400 个关键字。可以在学习过程中，把常用的、重要的关键字进行不断的积累和使用。所有关键字有自己特定的含义，尽量避免作为标识符。

2.1.2 MySQL 字符集的转换过程

编译 MySQL 时，系统默认的字符集是 latin1。可以通过如下方法进行转换。

(1) 最简单的修改方法，就是修改 MySQL 的 my.ini(C:\Program Files\MySQL\MySQL Server 5.7) 文件中的字符集，查找[mysql]键值，在下面加上一行“default-character-set=utf8”，如图 2-2 所示。



图 2-2 字符集的转换

修改完后，重启 MySQL 的服务，使用下列语句查看，发现数据库编码均已改成 UTF-8。

```
mysql > show variables like 'character % '
```

(2) 还有一种修改字符集的方法，就是使用 MySQL 的命令。用命令行的方式修改，只是临时更改，当服务器重启后，又将恢复默认设置。

```
mysql > set character_set_client = utf8;
mysql > set character_set_connection = utf8;
mysql > set character_set_database = utf8;
mysql > set character_set_results = utf8;
mysql > set character_set_server = utf8;
```

(3) 如果设置表的 MySQL 默认字符集为 UTF-8，并且通过 UTF-8 编码发送查询，有时存入数据库的仍然是乱码。问题就出在这个 connection 连接层上。

解决方法是在发送查询前执行一下下面这个语句：

```
MySQL > set names ('UTF8')
```

与这三个语句等价：

```
mysql> set character_set_client = (UTF8);
mysql> set character_set_results = (UTF8);
mysql> set character_set_connection = (UTF8);
```

2.1.3 MySQL 中的字符集层次设置

MySQL 对于字符集的支持细化到 4 个层次：服务器（Server）、数据库（DataBase）、数据表（Table）和连接（Connection）。

MySQL 对于字符集的指定可以细化到一个数据库、一张表和一列，并可以细化到应该用什么字符集。

MySQL 用下列的系统变量描述字符集。

（1）character_set_server 和 collation_server：这两个变量是服务器的字符集，默认的内部操作字符集。

（2）character_set_client：客户端来源数据使用的字符集，这个变量用来决定 MySQL 怎么解释客户端发到服务器的 SQL 命令文字。

（3）character_set_connection 和 collation_connection：连接层字符集。这两个变量用来决定 MySQL 怎么处理客户端发来的 SQL 命令。

（4）character_set_results：查询结果字符集，当 SQL 有结果返回的时候，这个变量用来决定发给客户端的结果中文字量的编码。

（5）character_set_database 和 collation_database：当前选中数据库的默认字符集，create database 命令有两个参数可以用来设置数据库的字符集和比较规则。

（6）character_set_system：系统元数据的字符集，数据库、表和列的定义都是用的这个字符集。它有一个定值，是 UTF-8。

对于以“collation_”开头的同上面对应的变量，用来描述字符集校对原则。

表的字符集：create table 的参数里可以设置，为列的字符集提供默认值。

列的字符集：决定本列的文字数据的存储编码。列的比较规则比 collation_connection 高。也就是说，MySQL 会把 SQL 中的文字直接转成列的字符集后再与列的文字数据比较。

字符集的依存关系如图 2-3 所示。

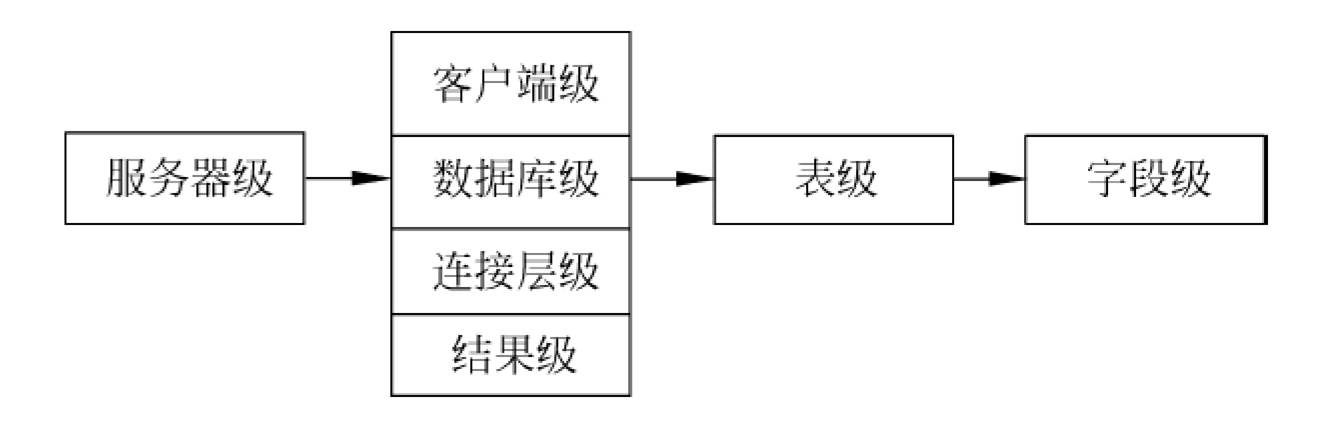


图 2-3　字符集的依附关系

- MySQL 默认的服务器级的字符集决定客户端、连接级和结果级的字符集。
- 服务器级的字符集决定数据库、客户端、连接层和结果集的字符集。
- 数据库的字符集决定表的字符集。
- 表的字符集决定字段的字符集。

2.1.4 常量和变量

1. 常量

常量也称为文字值或标量值，是指某个过程中值始终不变的量。MySQL 的常量类型和用法如表 2-1 所示。

表 2-1 MySQL 的常量类型

常量类型	常量表示说明	示 例
字符串	包括在单引号('')或双引号("")中，由字母(a～z、A～Z)、数字字符(0～9)以及特殊字符(如感叹号(!)、at 符(@)和数字号(#))组成	'China'、"Output X is:"、N'hello'(Unicode 字符串常量，只能用单引号括起字符串)
十进制整型	使用不带小数点的十进制数据表示	1234、654、+2008、-123
十六进制整型	使用前缀 0x 后跟十六进制数字串表示	0x1F00、0xEEC、0X19
日期	使用单引号('')将日期时间字符串括起来。MySQL 是按年-月-日的顺序表示日期的。中间的间隔符可以用"-"，也可以使用如"\""/""@"或"%"等特殊符号	'2018-01-03''2018/01/09''2017@12@10'
实型	有定点表示和浮点表示两种方式	897.1、-123.03、19E24、-83E2
位字段值	使用 b'value'符号写位字段值。value 是一个用 0 和 1 写成的二进制值。直接显示 b'value'的值可能是一系列特殊的符号	b'0'显示为空白，b'1'显示为一个笑脸图标
布尔	布尔常量只包含两个可能的值：true 和 false。false 的数字值为 0，true 的数字值为 1	获取 true 和 false 的值：select true，false
null 值	null 值可适用于各种列类型，它通常用来表示"没有值""无数据"等意义，并且不同于数字类型的"0"或字符串类型的空字符串	null

2. 系统变量

变量就是在某个过程中，其值是可以改变的量。可以利用变量存储程序执行过程中涉及的数据，如计算结果、用户输入的字符串以及对象的状态等。系统变量包括全局系统变量和会话系统变量两种类型。

(1) 全局变量和会话变量的区别：全局变量在 MySQL 启动时由服务器自动将它们初始化为默认值，主要影响整个 MySQL 实例的全局设置，大部分全局变量都是作为 MySQL 的服务器调节参数存在。对全局变量的修改会影响到整个服务器。会话变量在每次建立一个新的连接时，由 MySQL 来初始化。会话变量的定义是前面加一个@符号，随时定义和使用，会话结束就释放。即对会话变量的修改，只会影响到当前的会话，也就是当前的数据库连接。

(2) 大多数的系统变量应用于其他 SQL 语句时，必须在名称前加两个@符号。例如：

```
select @@version,current_date;
```

(3) 显示系统变量清单的格式。

```
show [global|session] variables [like '字符串']
```

例如查看字符"a"开头的系统变量命令如下：

```
show variables like 'a%'
```

(4) 修改系统变量的值。在 MySQL 中,有的系统变量的值是不能改变的,如@@version 和系统日期,而有些系统变量是可以通过 set 语句来修改的,例如将全局系统变量 sort_buffer_size 的值改为 25 000:

```
set @@global.sort_buffer_size = 25000;
```

再如:对于当前会话,把系统变量 sql_select_limit 的值设置为 100:

```
set @@session.sql_select_limit = 100;
```

该变量决定了 select 语句的结果集中的最大行数。执行如下命令可以显示:

```
select @@local.sql_select_limit;
```

也可以将一个系统变量值设置为 MySQL 默认值,可以使用 default 关键字。例如:

```
set @@local.sql_select_limit = default;
```

当然,用户也可以定义编程过程中自己需要的变量,此内容在后续章节中介绍。

2.2 MySQL 的数据类型

数据类型是数据的一种属性,其可以决定数据的存储格式、有效范围和相应的值范围限制。MySQL 的数据类型包括字符串类型、整数类型、浮点数类型、定点数类型、日期和时间类型以及二进制类型。在 MySQL 中创建表时,需要考虑为字段选择哪种数据类型是最合适的。选择了合适的数据类型,会提高数据库的效率。

2.2.1 字符串类型

字符串类型是在数据库中存储字符串的数据类型。字符串类型包括 char、varchar、blob、text、enum 和 set。

字符串类型可以分为两类:普通的文本字符串类型(char 和 varchar)和特殊类型(set 和 enum)。它们之间都有一定的区别,取值的范围不同,应用的地方也不同。

(1) 普通的文本字符串类型,即 char 和 varchar 类型,char 列的长度被固定为创建表所声明的长度,取值范围为 1~255;varchar 列的值是变长的字符串,取值和 char 一样。下面介绍普通的文本字符串类型如表 2-2 所示。

表 2-2 常规字符串类型

类　　型	取值范围	说　　明
[national] char(m) [binary|ASCII|unicode]	0~255 个字符	固定长度为 m 的字符串,其中 m 的取值范围为 0~255。National 关键字指定了应该使用的默认字符集。Binary 关键字指定了数据是否区分大小写(默认是区分大小写的)。ASCII 关键字指定了在该列中使用 latin1 字符集。Unicode 关键字指定了使用 UCS 字符集
char	0~255 个字符	char(m)类似
[national] varchar(m) [binary]	0~255 个字符	长度可变,其他和 char(m)类似

(2) 特殊类型 set 和 enum。特殊类型 set 和 enum 的介绍如表 2-3 所示。

表 2-3 enum 和 set 类型

类　　型	最大值	说　　明
Enum("value1","value2",…)	65 535	该类型的列只可以容纳所列值之一或为 null
Set("value1","value2",…)	64	该类型的列可以容纳一组值或为 null

说明：在创建表时，使用字符串类型时应遵循以下原则：

(1) 从速度方面考虑，要选择固定的列，可以使用 char 类型。

(2) 要节省空间，使用动态的列，可以使用 varchar 类型。

(3) 要将列中的内容限制在一种选择内，可以使用 enum 类型。

(4) 允许在一个列中有多于一个的条目，可以使用 set 类型。

(5) 如果要搜索的内容不区分大小写，可以使用 text 类型。

2.2.2 数字类型

数字类型总体可以分成整数类型和小数类型两类，小数类型又可以分为浮点数类型和定点数类型。

(1) 整数类型。整数类型是数据库中最基本的数据类型。标准 SQL 中支持 integer 和 smallint 这两类整数类型。MySQL 数据库除了支持这两种类型以外，还扩展支持了 tinyint、mediumint 和 bigint。MySQL 支持所有的 ANSI/ISO SQL 92 数字类型。这些类型包括准确数字的数据类型(numeric、decimal、integer 和 smallint)，还包括近似数字的数据类型(float、real 和 double precision)。其中的关键词 int 是 integer 的同义词，详细内容如表 2-4 所示。

表 2-4 整数类型

类　　型	取 值 范 围	说　　明	单位
tinyint	符号值：－127～127；无符号值：0～255	最小的整数	1 字节
bit	符号值：－127～127；无符号值：0～255	最小的整数	1 字节
bool	符号值：－127～127；无符号值：0～255	最小的整数	1 字节
smallint	符号值：－32 768～32 767 无符号值：0～65 535	小型整数	2 字节
mediumint	符号值：－8 388 608～8 388 607 无符号值：0～16 777 215	中型整数	3 字节
int	符号值：－2 147 683 648～2 147 683 647 无符号值：0～4 294 967 295	标准整数	4 字节
bigint	符号值：－9 223 372 036 854 775 808～9 223 372 036 854 775 807 无符号值：0～18 446 744 073 709 551 615	大整数	8 字节

(2) 小数类型。MySQL 中使用浮点数类型和定点数类型来表示小数。浮点数类型包括单精度浮点数(float)和双精度浮点数(double)。定点数类型就是 decimal，关键词 dec 是 decimal 的同义词，详细内容如表 2-5 所示。

表 2-5　小数类型

类　　型	取 值 范 围	说　　明	单　　位
float	＋(－)3.402823466E＋38	单精度浮点数	8 或 4 字节
double	＋(－)1.7976931348623157E＋308 ＋(－)2.2250738585072014E－308	双精度浮点数	8 字节
decimal	可变	定点小数	自定义长度

说明：在创建表时，使用哪种数字类型，应遵循以下原则。

(1) 选择最小的可用类型，如果值永远不超过 127，则使用 tinyint 比 int 强。

(2) 对于完全都是数字的，可以选择整数类型。

(3) 浮点数类型用于可能具有小数部分的数。例如货物单价、网上购物交付金额等。

2.2.3　日期和时间类型

日期与时间类型是为了方便在数据库中存储日期和时间而设计的。MySQL 中有多种表示日期和时间的数据类型。其中，year 类型表示年份；date 类型表示日期；time 类型表示时间；datetime 和 timestamp 表示日期和时间。其中的每种类型都有其取值的范围，如赋予它一个不合法的值，将会被“0”代替。下面介绍日期和时间类型，如表 2-6 所示。

表 2-6　日期和时间类型

类　　型	取 值 范 围	说　　明
date	1000-01-01～9999-12-31	日期，格式 YYYY-MM-DD
time	－838:58:59　835:59:59	时间，格式 HH：MM：SS
datetime	1000-01-01 00:00:00 9999-12-31 23:59:59	日期和时间，格式 YYYY-MM-DD HH：MM：SS
timestamp	1970-01-01 00:00:00 2037 年的某个时间	时间标签，在处理报告时使用显示格式取决于 M 的值
year	1901—2155	年份可指定两位数字和四位数字的格式

2.2.4　二进制类型

二进制类型是在数据库中存储二进制数据的类型。二进制类型包括 binary、varbinary、bit、tinyblob、blob、mediumblob 和 longblob 类型。tinytext、longtext 和 text 等适合存储长文本的类型，也放在这里介绍。

其中，text 和 blob 类型的大小可以改变，text 类型适合存储长文本，而 blob 类型适合存储二进制数据，支持任何数据，例如文本、声音和图像等。text 和 blob 类型如表 2-7 所示。

表 2-7　text 和 blob 类型

类　　型	最大长度(字节数)	说　　明
tinyblob	2^8-1(225)	小 blob 字段
tinytext	2^8-1(225)	小 text 字段
blob	$2^{16}-1$(65 535)	常规 blob 字段
text	$2^{16}-1$(65 535)	常规 text 字段

续表

类　　型	最大长度(字节数)	说　　明
mediumblob	$2^{24}-1$(16 777 215)	中型 blob 字段
mediumtext	$2^{24}-1$(16 777 215)	中型 text 字段
longblob	$2^{32}-1$(4 294 967 295)	长 blob 字段
longtext	$2^{32}-1$(4 294 967 295)	长 text 字段

2.3　MySQL 的运算符和表达式

运算符是用来连接表达式中各个操作数的符号,其作用是指明对操作数所进行的运算。MySQL 数据库通过使用运算符,不但可以使数据库的功能更加强大,而且可以更加灵活地使用表中的数据。MySQL 运算符包括 4 类,分别是算术运算符、比较运算符、逻辑运算符和位运算符。

需要说明的是:MySQL 中的 select 语句具有输出功能,能够显示函数和表达式的值。

2.3.1　算术运算符

算术运算符是 MySQL 中最常用的一类运算符。MySQL 支持的算术运算符包括:加、减、乘、除、求余。下面列出算术运算符的符号和作用,如表 2-8 所示。


算术运算符

表 2-8　算术运算符

符号	作　　用	符号	作　　用
+	加法运算	%	求余运算
−	减法运算	div	除法运算,返回商,同“/”
*	乘法运算	mod	求余运算,返回余数,同“%”
/	除法运算		

说明:加(+)、减(−)和乘(*)可以同时运算多个操作数。除号(/)和求余运算符(%)也可以同时计算多个操作数,但是这两个符号计算多个操作数不太好。div 和 mod 这两个运算符只有两个参数。进行除法和求余的运算时,除以零的除法是不允许的,MySQL 会返回 null。运算符 div 的运算结果是整数。

【例 2-1】 使用算术运算符进行加、减、乘、除、求余等运算。

代码和运算结果如下:

```
mysql > select 3 + 2,1.5 * 3,3/5,100 - 23.5,5 % 3;
        +----+------+-------+---------+-----+
        | 3 + 2 | 1.5 * 3 | 3/5    | 100 - 23.5 | 5 % 3 |
        +----+------+-------+---------+-----+
        |  5  |  4.5  | 0.6000 |      76.5 |    2 |
        +----+------+-------+---------+-----+
        1 row in set (0.01 sec)
```

2.3.2 比较运算符


比较运算符

比较运算符是查询数据时最常用的一类运算符。select 语句中的条件语句经常要使用比较运算符。通过这些比较运算符,可以判断表中的哪些记录是符合条件的。比较运算符的符号、名称和应用示例如表 2-9 所示。

表 2-9 比较运算符

运算符	作用	示例	运算符	作用	示例
=	等于	Id=5	is not null	非空	Id is not null
>	大于	Id>5	between	区间比较	Id between 1 and 15
<	小于	Id<5	in	属于	Id in (3,4,5)
>=	大于或等于	Id>=5	not in	不属于	Name not in (shi,li)
<=	小于或等于	Id<=5	like	模式匹配	Name like ('shi%')
!=或<>	不等于	Id!=5	not like	模式匹配	Name not like ('shi%')
Is null	空	Id is null	regexp	常规表达式	Name regexp 正则表达式

下面对几种较常用的比较运算符进行详解。

(1) 运算符“=”。“=”用来判断数字、字符串和表达式等是否相等。如果相等,返回 1,否则返回 0。

说明:在运用“=”运算符判断两个字符是否相同时,数据库系统都是根据字符的 ASCII 码进行判断的。如果 ASCII 码相等,则表示这两个字符相同。如果 ASCII 码不相等,则表示两个字符不同。空值(null)不能使用“=”来判断。

(2) 运算符“<>”和“!=”。“<>”和“!=”用来判断数字、字符串、表达式等是否不相等。如果不相等,则返回 1;否则,返回 0。这两个符号也不能用来判断空值(null)。

(3) 运算符“>”。“>”用来判断左边的操作数是否大于右边的操作数。如果大于,返回 1;否则,返回 0。同样空值(null)不能使用“>”来判断。“<”运算符、“<=”运算符和“>=”运算符都与“>”运算符使用方法基本相同,这里不再赘述。

【例 2-2】 使用比较运算符进行判断运算。

代码和运算结果如下:

```
mysql> select 'A'>'B',1+1=2, 'X'<'x', 7<>7, 'a'<= 'a';
        +--------+-------+--------+------+---------+
        | 'A'>'B' |1+1=2|  'X'<'x' |  7<>7 | 'a'<= 'a' |
        +--------+-------+--------+------+---------+
        |    0    |      1|       0 |     0 |       1    |
        +--------+-------+--------+------+---------+
        1 row in set (0.02 sec)
```

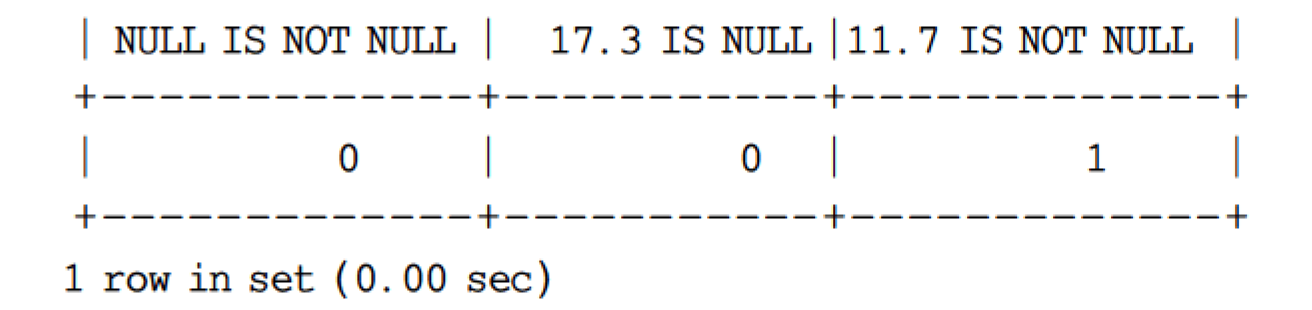
(4) 运算符“is null”。“is null”用来判断操作数是否为空值(null)。操作数为 null 时,结果返回 1;否则,返回 0。is not null 刚好与 is null 相反。


空运算符

【例 2-3】 运用 is null、is not null 运算符的用法。

代码和运算结果如下:

```
mysql> select null is not null,17.3 is null, 11.7 is not null;
        +-------------+-----------+-------------+
```



```
| NULL IS NOT NULL |  17.3 IS NULL |11.7 IS NOT NULL  |
+------------------+--------------+------------------+
|                0 |            0 |                1 |
+------------------+--------------+------------------+
1 row in set (0.00 sec)
```

说明：

① “=”“<>”“!=”“>”“>=”“<”“<=”等运算符都不能用来判断空值(null)。一旦使用，结果将返回 null。如果要判断一个值是否为空值，可以使用“<>”、is null 和 is notnull 来判断。

② null 和'null'是不同的，前者表示为空值，后者表示一个由 4 个字母组成的字符串。



between and
运算符

(5) 运算符“between and”。“between and”用于判断数据是否在某个取值范围内。也可以添加 not 运算符对一个 between 运算进行取反。其表达式如下：

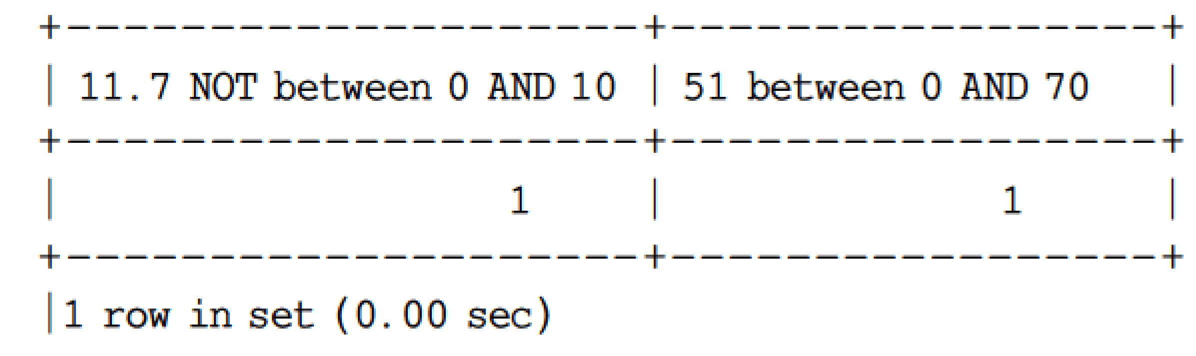
```
x1 between m and n
```

如果 x1 大于等于 m，且小于等于 n，结果将返回 1，否则将返回 0。

【例 2-4】 运用“between and”运算符判断一个数是否在某个范围。

代码和运算结果如下：

```
mysql> select 11.7 not between 0 and 10, 51 between 0 and 70;
```



```
+---------------------------+---------------------+
| 11.7 NOT between 0 AND 10 | 51 between 0 AND 70 |
+---------------------------+---------------------+
|                         1 |                   1 |
+---------------------------+---------------------+
|1 row in set (0.00 sec)
```

(6) 运算符“in”。“in”用于判断数据是否存在于某个集合中。其表达式如下：

```
x1 in(值 1,值 2,…,值 n)
```



in 运算符

如果 x1 等于值 1 到值 n 中的任何一个值，结果将返回 1。如果不是，结果将返回 0。

【例 2-5】 运用“in”运算符判断某值是否在指定的范围内。

代码和运算结果如下：

```
mysql> select 7 in(1,2,5,6,7,8,9), 3 notin (1,10);
        +---------------------+----------------+
        | 7 IN(1,2,5,6,7,8,9) | 3 NOT IN (1,10) |
        +---------------------+----------------+
        |                   1 |              1 |
        +---------------------+----------------+
        1 row in set (0.00 sec)
```

(7) 运算符“like”。“like”用来匹配字符串。其中，“%”匹配任意个字符，“_”匹配一个字符。其表达式如下：

```
x1 like s1
```


like 运算符

如果 x1 与字符串 s1 匹配，结果将返回 1。否则返回 0。

【例 2-6】 使用 like 运算符，判断某字符串是否与指定的字符串匹配。代码和运算结果如下：

```
mysql > select 'MySQL' like 'MY%', 'APPLE' like 'A_';
        +-----------------+---------------+
        | 'MySQL' like 'MY%' | 'APPLE' like 'A_' |
        +-----------------+---------------+
        |        1        |       0       |
        +-----------------+---------------+
        1 row in set (0.00 sec)
```

(8) 运算符 regexp。regexp 同样用于匹配字符串，但其使用的是正则表达式进行匹配。使用 regexp 运算符匹配字符串，其使用方法非常简单。regexp 运算符经常与“^”“$”和“.”一起使用。“^”用来匹配字符串的开始部分；“$”用来匹配字符串的结尾部分；“.”用来代表字符串中的一个字符。其表达式格式如下：

```
x1 regexp '匹配方式'
```

如果 x1 满足匹配方式，结果将返回 1；否则将返回 0。

regexp 运算符一般用来与表中的字段值匹配，判定该值是否以指定字符开头、中间的一个字符或结尾，同时判定是否包含指定的字符串，具体内容在以后的相关章节中会介绍实例。

2.3.3 逻辑运算符


逻辑运算符

逻辑运算符用来判断表达式的真假。如果表达式是真，结果返回 1。如果表达式是假，结果返回 0。逻辑运算符又称为布尔运算符。MySQL 中支持 4 种逻辑运算符，分别是与、或、非和异或。下面是 4 种逻辑运算符的符号及作用，如表 2-10 所示。

表 2-10 逻辑运算符

逻辑运算符	作　　用	逻辑运算符	作　　用
&& 或 and	与	! 或 not	非
|| 或 or	或	xor	异或

(1) 与运算。“&&”或者“and”是与运算的两种表达方式。如果所有数据不为 0 且不为空值(null)，结果返回 1；如果存在任何一个数据为 0，结果返回 0；如果存在一个数据为 null 且没有数据为 0，结果返回 null。与运算符支持多个数据同时进行运算。

(2) 或运算。“||”或者“or”表示或运算。所有数据中存在任何一个数据不为非 0 的数字，结果返回 1；如果数据中不包含非 0 的数字，但包含 null 时，结果返回 null；如果操作数中只有 0，结果返回 0。或运算符“||”也可以同时操作多个数据。

(3) 非运算。“!”或者 not 表示非运算。通过非运算，将返回与操作数据相反的结果。如果操作数据是非 0 的数字，结果返回 0；如果操作数据是 0，结果返回 1；如果操作数据是 null，结果返回 null。

【例 2-7】 逻辑运算符 and、or、not 示例。

代码和运算结果如下：

```
mysql>select not('A' = 'B'),('c' = 'C')and('c'<'D')or(1 = 2);
        +------------+--------------------------+
        | NOT('A' = 'B') | ('c' = 'C') AND ('c'<'D') OR(1 = 2) |
        +------------+--------------------------+
        |          1   |                         1    |
        +------------+--------------------------+
        1 row in set (0.00 sec)
```

(4) 异或运算。xor 表示异或运算。只要其中任何一个操作数据为 null，结果返回 null；如果两个操作数都是非 0 值或者 0 值，则返回结果为 0；如果一个为 0 值，另一个为非 0 值，则返回结果为 1。

【例 2-8】 逻辑运算符 &&、xor 示例。

代码和运算结果如下：

```
mysql> select ('c'= 'C') && (1 = 2),('A' = 'a')xor(1 + 1 = 3);
        +-----------------+-----------------+
        | ('c'= 'C') && (1 = 2)    | ('A' = 'a')XOR(1 + 1 = 3) |
        +-----------------+-----------------+
        |              0           |                1     |
        +-----------------+-----------------+
        1 row in set (0.02 sec)
```

2.3.4 位运算符

位运算符是在二进制数上进行计算的运算符。位运算会先将操作数变成二进制数，进行位运算。然后再将计算结果从二进制数变回十进制数。MySQL 中支持 6 种位运算符。分别是：按位与、按位或、按位取反、按位异或、按位左移和按位右移。6 种位运算符的符号及作用如表 2-11 所示。



位运算符

表 2-11 位运算符

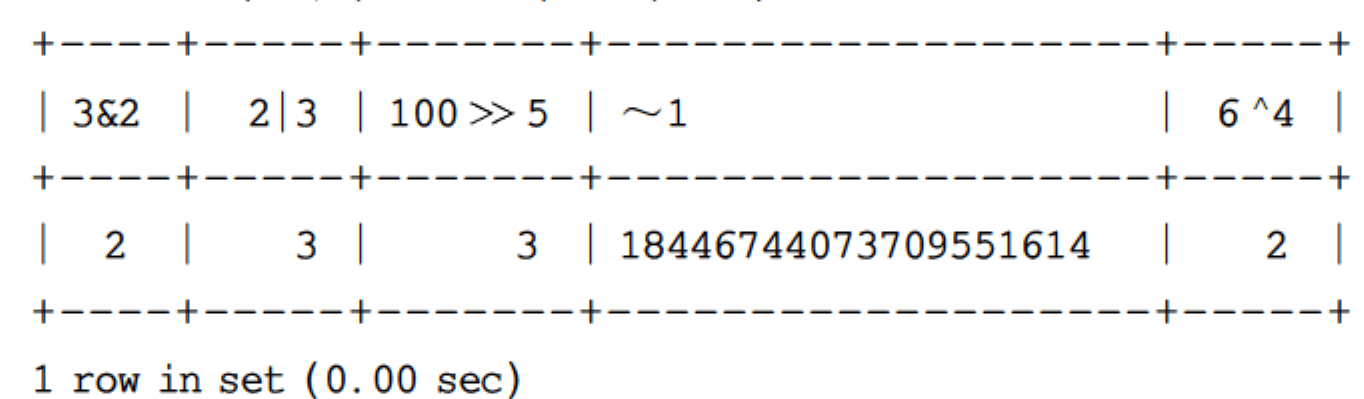
符号	作　用
&	按位与。进行该运算时，数据库系统会先将十进制的数转换为二进制的数。然后对应操作数的每个二进制位上进行与运算。1 和 1 相与得 1，与 0 相与得 0。运算完成后再将二进制数变回十进制数
|	按位或。将操作数化为二进制数后，每位都进行或运算。1 和任何数进行或运算的结果都是 1，0 与 0 或运算结果为 0
~	按位取反。将操作数化为二进制数后，每位都进行取反运算。1 取反后变成 0，0 取反后变成 1
^	按位异或。将操作数化为二进制数后，每位都进行异或运算。相同的数异或之后结果是 0，不同的数异或之后结果为 1
<<	按位左移。“m<<n”表示 m 的二进制数向左移 n 位，右边补上 n 个 0。例如，二进制数 001 左移 1 位后将变成 0010
>>	按位右移。“m>>n”表示 m 的二进制数向右移 n 位，左边补上 n 个 0。例如，二进制数 011 右移 1 位后变成 001，最后一个 1 直接被移出

位运算符是在二进制数上进行计算的运算符。位运算会先将操作数变成二进制数，然后进行位运算。再将计算结果从二进制数变回十进制数。

【例 2-9】 位运算符示例。

代码和运算结果如下：

```
mysql > select 3&2,2 |3,100 >> 5, ~1,6 ^4;
```



```
+----+-----+-------+--------------------+-----+
| 3&2 |  2|3 | 100 >> 5 | ~1                 |  6 ^4 |
+----+-----+-------+--------------------+-----+
|  2  |    3 |        3 | 18446744073709551614 |    2 |
+----+-----+-------+--------------------+-----+
1 row in set (0.00 sec)
```

2.3.5 表达式和运算符的优先级

1. 表达式

在 SQL 语言中，表达式就是常量、变量、列名、复杂计算、运算符和函数的组合。一个表达式通常都有返回值。与常量和变量一样，表达式的值也具有某种数据类型。根据表达式的值的类型，表达式可分为字符型表达式、数值型表达式和日期型表达式。

2. 运算符的优先级

当一个复杂的表达式有多个运算符时，运算符优先级决定执行运算的先后次序。在一个表达式中按先高(优先级数字小)后低(优先级数字大)的顺序进行运算。运算符优先级如表 2-12 所示。按照从高到低，从左到右的级别进行运算操作。如果优先级相同，则表达式左边的运算符先运算。

表 2-12 MySQL 运算符的优先级

优先级	运　算　符
1	!
2	～
3	^
4	＊,/,div,%,mod
5	＋,－
6	＞＞,＜＜
7	&
8	|
9	＝,＜＝＞,＜,＜＝,＞,＞＝,!＝,＜＞,in,is,null,like,regexp
10	between and,case,when,then,else
11	not
12	&&,and
13	||,or,xor
14	:＝ (赋值号)

2.4 MySQL 的常用函数

MySQL 数据库中提供了很丰富的函数。这些内部函数可以帮助用户更加方便地处理表中的数据。MySQL 函数包括数学函数、字符串函数、日期和时间函数、条件判断函数、系统信息函数、加密函数和格式化函数等。

select 语句及其条件表达式都可以使用这些函数。同时，insert、update 和 delete 语句及其条件表达式也可以使用这些函数。

2.4.1 数学函数



数学函数

数学函数是 MySQL 中常用的一类函数。主要用于处理数字，包括整型、浮点数等。数学函数包括绝对值函数、正弦函数、余弦函数、获取随机数的函数等，常用的数学函数如表 2-13 所示。

表 2-13 常用数学函数

函　数	功能描述
abs	返回表达式的绝对值
acos	反余弦函数，返回以弧度表示的角度值
asin	反正弦函数，返回以弧度表示的角度值
atan	反正切函数，返回以弧度表示的角度值
ceiling	返回大于或等于指定数值表达式的最小整数
cos	返回以弧度为单位的角度的余弦值
degree	弧度值转换为角度值
exp	返回给定表达式为指数的 e 值
floor	返回小于或等于指定数值表达式的最大整数
greatest	获得一组数中的最大值
least	获得一组数中的最小值
log	返回给定表达式的自然对数
log10	返回给定表达式的以 10 为底的对数
PI	常量，圆周率
pow	返回给定表达式的指定次方的值
radians	角度值转换为弧度值
rand	返回 0～1 之间的随机 float 数
round	返回指定小数的位数的表达式的值
sign	返回某个数的符号
sin	返回以弧度为单位的角度的正弦值
sqrt	返回给定表达式的平方根
tan	返回以弧度为单位的角度的正切值

数学函数可以作为表达式或表达式的一部分使用，下面举几个例子。

【例 2-10】 floor()、ceiling()和 log()函数示例。

代码和运算结果如下：

```
mysql >select floor(3.67),ceiling(4.71), log(5);
```

```
+-----------+------------+-----------------+
| floor(3.67) | ceiling(4.71) |   log(5)        |
+-----------+------------+-----------------+
|         3 |          5 | 1.6094379124341003 |
+-----------+------------+-----------------+
1 row in set (0.05 sec)
```

【例 2-11】 利用取随机数 rand()和四舍五入 round()输出 60～90 和 25～65 之间的任意整数。

代码和运算结果如下：

```
mysql>select 60 + round(30 * rand(),0) , 25 + round(40 * rand(),0);
+--------------------+----------------------+
| 60 + round(30 * rand(),0) | 25 + round(40 * rand(),0)  |
+--------------------+----------------------+
|                 76  |                  39  |
+--------------------+----------------------+
1 row in set (0.00 sec)
```

说明：

(1) rand()返回的数是完全随机的，而 rand(x)函数的 x 相同时，它被用作种子值，返回的值是相同的。

(2) 四舍五入 round(x)返回离 x 最近的整数，即对 x 进行四舍五入处理。round(x,y)返回 x 保留到小数点后 y 位的值，在截取时进行四舍五入处理。

2.4.2 字符串函数



字符串函数

字符串函数主要用于处理字符串数据和表达式，MySQL 中的字符串函数包括计算字符串长度函数、合并函数、替换函数、比较函数和查找字符串位置函数等，常用字符串函数及其功能如表 2-14 所示。

表 2-14 常用字符串函数

函数名称	功能描述
char_length	返回字符串中字符的个数
concat	返回连接参数产生的字符串
left	返回从字符串左边开始指定个数的字符
length	返回给定字符串字节长度
lower	将大写字符数据转换为小写字符数据后返回字符表达式
ltrim	删除起始空格后返回字符表达式
replace	用第三个表达式替换第一个字符串表达式中，出现的所有第二个给定字符串表达式
repeat	指定字符串和重复连接次数，返回由其连接而成的字符串
reverse	返回字符表达式的反转
right	返回从字符串右边开始指定个数的字符
rtrim	截断所有尾随空格后返回一个字符串
space	返回由重复的空格组成的字符串
substring	求子串函数
upper	返回将小写字符数据转换为大写的字符表达式

下面就常用字符串函数举例说明。

【例 2-12】 利用 concat()函数连接字符串。

代码和运算结果如下：

```
mysql> select concat('MY','SQL','5.7'),concat('ABC',null,'DEF');
        +----------------------+---------------------+
        | concat('MY','SQL','5.7')  | concat('ABC', null,'DEF') |
        +----------------------+---------------------+
        | MYSQL5.7               | null                |
        +----------------------+---------------------+
        1 row in set (0.04 sec)
```

说明：concat()函数返回来自于参数连接的字符串。如果任何参数是 null，返回 null。可以有超过两个的参数。数字参数会被变换为等价的字符串形式。

【例 2-13】 利用 substring()函数返回指定字符串，并利用 reverse()逆序输出。

代码和运算结果如下：

```
mysql> select substring('ABCDEFGH',2,6),
    -> reverse(substring('ABCDEFGH',2,6));
      +----------------------+----------------------------+
      | substring('ABCDEFGH',2,6)  | reverse(substring('ABCDEFGH',2,6)) |
      +----------------------+----------------------------+
      | BCDEFG               | GFEDCB                     |
      +----------------------+----------------------------+
      1 row in set (0.00 sec)
```

2.4.3 日期和时间函数



日期和时间函数

日期和时间函数主要用于处理表中的日期和时间数据。日期和时间函数包括获取当前日期的函数、获取当前时间的函数、计算日期的函数和计算时间的函数等，常用日期时间函数如表 2-15 所示。

表 2-15 常用日期时间函数

函 数 名	功 能 描 述
curdate	获取当前系统的日期
curtime	获取当前系统的时间
date_add	可以对日期和时间进行加法运算
date_sub	可以对日期和时间进行减法运算
datediff	计算两个日期相隔的天数
date_format	用来格式化日期值
day	获取指定日期的日期整数
dayname	以英文名方式显示，返回指定日期是星期几，如 Tuesday 等
dayofmonth	返回指定日期在一个月中的序数
dayofweek	返回指定日期在一个星期中的序数
dayofyear	返回指定日期在一年中的序数
hour	返回指定时间的小时数

续表

函　数　名	功 能 描 述
minute	返回指定时间的分钟数
month	获取指定日期的月份整数
now/sysdayte	返回当前日期和时间
quarter	获取指定日期的季度整数
second	返回指定时间的秒钟数
time_format	用来格式化时间值
UTC_DATE	用来输出世界标准时间的日期
UTC_TIME	用来输出世界标准时间
year	获取指定日期的年份整数

(1) 常用日期时间函数举例。

【例 2-14】 利用 curdate()和 curtime()函数返回当前日期和时间。

代码和运算结果如下:

```
mysql >select curdate(),curtime();
       +-----------+---------+
       | curdate() | curtime() |
       +-----------+---------+
       | 2017 - 04 - 24 | 08:33:59 |
       +-----------+---------+
       1 row in set (0.05 sec)
```

【例 2-15】 返回指定日期在一年、一星期及一个月中的序数。

代码和运算结果如下:

```
mysql > select dayofyear(20170512),dayofmonth('2017 - 05 - 12'),
      > dayofweek(now());
      +-----------------+--------------------+--------------+
      | dayofyear(20170512) |dayofmonth('2017 - 05 - 12') |dayofweek(now()) |
      +-----------------+--------------------+--------------+
      |             132 |                 12 |            2 |
      +-----------------+--------------------+--------------+
      1 row in set (0.02 sec)
```

【例 2-16】 返回指定时间的小时、分钟、秒钟。

代码和运算结果如下:

```
mysql > select curtime(),hour(curtime()),
      > minute(curtime()),second(curtime());
      +---------+-------------+----------------+-----------------+
      | curtime() | hour(curtime()) | minute(curtime()) | second(curtime()) |
      +---------+-------------+----------------+-----------------+
      |08:39:52 |           8 |             39 |              52 |
      +---------+-------------+----------------+-----------------+
      1 row in set (0.00 sec)
```

（2）date_add()和 date_sub()函数可以对日期和时间进行算术操作，它们分别用来增加和减少日期值。date_add()和 date_sub()函数的语法格式为：



日期和时间的算术操作

```
date_add | date_sub(date, interval int keyword)
```

date 表示日期和时间，interval 关键字表示一个时间间隔。使用的关键字如表 2-16 所示。

表 2-16　date_add()和 date_sub()函数的参数

关　键　字	间隔值的格式	关　键　字	间隔值的格式
day	日期	minute	分钟
day_hour	日期：小时	minute_second	分钟：秒
day_minute	日期：小时：分钟	month	月
day_second	日期：小时：分钟：秒	second	秒
hour	小时	year	年
hour_minute	小时：分钟	year_month	年-月
hour_second	小时：分钟：秒		

【例 2-17】　计算指定时间的 45 分钟前是什么时间。

代码和运算结果如下：

```
mysql > select date_sub('2017 - 10 - 1 10:10:10', interval 45 minute);
        +-------------------------------------------+
        | date_sub('2017 - 10 - 1 10:10:10', interval 45 minute)   |
        +-------------------------------------------+
        | 2017 - 10 - 01 09:25:10                    |
        +-------------------------------------------+
        1 row in set (0.03 sec)
```

【例 2-18】　7 月 15 日放假，计算现在离放假还有多少天。

代码和运算结果如下：

```
mysql > select datediff('2017 - 7 - 15',now());
        +---------------------------+
        | datediff('2017 - 7 - 15',now())   |
        +---------------------------+
        |                        82 |
        +---------------------------+
        1 row in set (0.03 sec)
```

（3）日期和时间格式化的函数。date_format()和 time_format()函数可以用来格式化日期和时间值。

语法格式如下：

```
date_format/ time_format(date | time, fmt)
```

其中，date 和 time 是需要格式化的日期和时间值，fmt 是日期和时间值格式化的形式，表 2-17 列出了 MySQL 中的日期/时间格式化代码。

表 2-17　MySQL 中的日期/时间格式化代码

关键字	间隔值的格式	关键字	间隔值的格式
%a	缩写的星期名(Sun,Mon…)	%p	AM 或 PM
%b	缩写的月份名(Jan,Feb…)	%r	时间,12 小时的格式
%d	月份中的天数	%S	秒(00,01)
%H	小时(01,02…)	%T	时间,24 小时的格式
%I	分钟(00,01…)	%w	一周中的天数(0,1)
%j	一年中的天数(001,002…)	%W	长型星期的名字(Sunday,Monday…)
%m	月份,2 位(00,01…)	%Y	年份,4 位
%M	长型月份的名字(January,February)		

【例 2-19】 按照指定格式码输出日期。

代码和运算结果如下:

```
mysql> select date_format(now(),'%W, %d, %m,  %Y ,  %r,  %p');
      +-----------------------------------------+
      | date_format(now(),'%W, %d, %m,  %Y ,  %r,  %p')  |
      +-----------------------------------------+
      | Monday,24,04, 2017 , 08:53:25 AM, AM            |
      +-----------------------------------------+
      1 row in set (0.03 sec)
```

2.4.4　聚合函数

聚合函数也称为分组统计函数。这些函数的主要功能如表 2-18 所示。常用于对聚合在组内的数据表行进行计算。

表 2-18　MySQL 中的聚合函数

函　　数	功 能 描 述
avg	返回组中数据的平均值,忽略 null 值
count	返回组中项目的数量
max	返回多个数据比较的最大值,忽略 null 值
min	返回多个数据比较的最小值,忽略 null 值
sum	返回组中数据的和,忽略 null 值

2.4.5　其他函数

1. 系统信息函数

系统信息函数(表 2-19)用来查询 MySQL 数据库的系统信息。例如,查询数据库的版本,查询数据库的当前用户等。

表 2-19　MySQL 中的系统信息函数

函　　数	功 能 描 述
database()	返回当前数据库名
benchmark(n,expr)	将表达式 expr 重复运行 n 次
charset(str)	返回字符串 str 的字符集
connection_id()	返回当前客户连接服务器的次数
found_rows()	将最后一个 MySQL>select 查询(没有以 limit 语句进行限制)返回的记录行数返回
get_lock(str,dur)	获得一个由字符串 str 命名的并且有 dur 秒延时的锁定
is_free_lock(str)	检查以 str 命名的锁定是否释放
last_insert_id()	返回由系统自动产生的最后一个 autoincrement id 的值
master_pos_wait(log,pos,dur)	锁定主服务器 dur 秒直到从服务器与主服务器的日志 log 指定的位置 pos 同步
release_lock(str)	释放由字符串 str 命名的锁定
user()或 system_user()	返回当前登录用户名
version()	返回 MySQL 服务器的版本

【例 2-20】　返回 MySQL 服务器的版本、当前数据库名和当前用户名信息,并查看当前用户连接 MySQL 服务器的次数。

代码和运算结果如下:

```
mysql > select version(),database(),user(),connection_id();
      +----------+------------+-------------+-------------+
      | version() | database()  | user()       | CONNECTION_ID() |
      +----------+------------+-------------+-------------+
      | 5.7.17 - log | NULL       | root@localhost |          12 |
      +----------+------------+-------------+-------------+
      1 row in set (0.06 sec)
```

2. 加密函数

password(str)函数可以对字符串 str 进行加密。一般情况下,password(str)函数主要是用来给用户的密码加密的。

【例 2-21】　使用 password(str)函数对字符串“student1357”加密。

代码和运算结果如下:

```
mysql > select password('student1357');
      +-------------------------------------------+
      | password('student1357')                    |
      +-------------------------------------------+
      | *839D903605F413D56D2A0635F7A10D019E48785E  |
      +-------------------------------------------+
      1 row in set, 1 warning (0.00 sec)
```

例如,可以修改学生 Li Ping 的密码为 student1357,并加密:

```
mysql > set password for 'Li Ping'@'localhost' = password('student1357');
```

3. 格式化函数

format()函数语法格式为：

```
format(x, y)
```

format()函数把数值格式化为以逗号分隔的数字序列。format()的第一个参数 x 是被格式化的数据，第二个参数 y 是结果的小数位数。

【例 2-22】 利用格式函数 format()处理数据。

代码和运算结果如下：

```
mysql > select format(2/3,2), format(123456.78,0);
        +-------------+-----------------+
        | format(2/3,2)  | format(123456.78,0) |
        +-------------+-----------------+
        | 0.67          | 123,457            |
        +-------------+-----------------+
        1 row in set (0.02 sec)
```

用户根据工作需要，可以创建用户定义函数，还可以利用流程控制语句编写较为实用的程序，以提高程序开发和运行的质量，相关内容在以后的章节中介绍。

2.5 小 结

本章介绍了 MySQL 数据库常见的数据类型。整数类型、浮点数类型、日期和时间类型以及字符串类型是数据库中使用最频繁的数据类型。定点数类型、二进制类型使用相对比较少。在实际的应用过程中，可以根据不同的需要选择相应的数据类型。MySQL 的关键字很多、运算符特别丰富、函数种类多且功能强，应用很广泛。在学习过程中，可以采取边用边学的方法，逐步去掌握它们中的重点对象。学习本章后需要重点掌握如下内容：

- 构成 MySQL 语言的字符集和字符序。
- 关键字、常量、运算符、常用函数的用法。
- 系统变量的使用方法。
- 标识符、变量以及表达式的定义和使用方法。

习 题 2

1. 选择题

(1) 以下命令中，________是 DML 语句。

A. create　　B. alter　　C. select　　D. drop

(2) 以下关于 MySQL 的说法中错误的是________。

A. MySQL 是一种关系型数据库管理系统

B. MySQL 软件是一种开放源码软件

C. MySQL 服务器工作在客户端/服务器模式下或嵌入式系统中

D. 在 Windows 系统下书写 MySQL 语句区分大小写

(3) 控制台中执行________语句时可以退出 MySQL。

A. exit　　B. go 或 quit　　C. go 或 exit　　D. exit 或 quit

(4) 关于 MySQL 数据库的说法,选项________的说法是错误的。

A. MySQL 数据库不仅开放源码,而且能够跨平台使用。例如,可以在 Windows 操作系统中安装 MySQL 数据库,也可以在 Linux 操作系统中使用 MySQL 数据库

B. MySQL 数据库启动服务时有两种方式,如果服务已经启动可以在任务管理器中查找 mysqlId. exe 程序,如果该进程存在则表示正在运行

C. 手动更改 MySQL 的配置文件 my. ini 时,只能更改与客户端有关的配置,而不能更改与服务器端相关的配置信息

D. 登录 MySQL 数据库成功后,直接输入"help;"语句后,按 Enter 键可以查看帮助信息

(5) 下列________类型不是 MySQL 中常用的数据类型。

A. int　　B. var　　C. time　　D. char

(6) 在 MySQL 中会话变量前面的字符为________。

A. 空格　　B. #　　C. @@　　D. @

(7) 设置表的默认字符集关键字是________。

A. default　character　　B. default　set

C. default　　D. default character set

2. 简答题

(1) 简述字符集 UTF-8 和 GB 2312 的区别。

(2) datetime 类型和 timestamp 类型的相同点和不同点是什么?

(3) MySQL 支持的数据类型主要有哪几类? 17 属于什么类型?'17'属于什么类型?

(4) 简述系统变量、全局变量和会话变量的关系。

(5) 简述聚合函数的特点和用途。

3. 上机练习

(1) 创建文本文件。将查询系统当前日期、当前时间以及到 2019 年 1 月 1 号还有多少天。然后通过 MySQL 命令执行文本文件中的内容。

(2) 利用随机函数输出 20～90 的任意两个数(含 2 位小数)。

(3) 计算 1000 天后的日期和 3000 分钟后的日期时间。

第 3 章　MySQL 数据库的基本操作

前面学习了设计数据库的基本理论，在此基础上就可以完成以教务管理数据库为例的 MySQL 数据库的概念结构设计和逻辑结构设计部分，下一步就可以实现在 MySQL 的软件环境中创建和维护数据库的操作。利用 MySQL 或 MySQL Workbench 可视化软件创建并维护数据库。

本章将学习设计数据库的基本过程，以及创建和管理数据库的基本操作。

3.1　MySQL 数据库概述

MySQL 数据库的管理主要包括数据库的创建、打开当前数据库、显示数据库结构以及删除数据库等操作。

MySQL 数据库管理系统提供了许多命令行工具，这些工具可以用来管理 MySQL 服务器、对数据库进行访问控制、管理 MySQL 用户以及备份和恢复数据库等。MySQL 也提供图形化管理工具，这使得对数据库的操作更加简单。

MySQL 数据库主要有以下特点：

- 可移植性好；
- 具有扩展性和灵活性；
- 具有强大的数据保护功能；
- 能够支持大型数据库；
- 具有超强的稳定性；
- 具有强大的查询功能。

3.1.1　MySQL 数据库文件

数据库管理的主要任务包括创建、操作和支持数据库。在 MySQL 中，每个数据库都对应存放在一个与数据库同名的文件夹中。MySQL 数据库文件有. frm、. myd 和. myi 三种文件，其中. frm 是描述表结构的文件，. myd 是表的数据文件，. myi 是表数据文件中的索引文件。它们都存放在与数据库同名的文件夹中。数据库的默认存放位置是 C:\Documents and Settings\All Users\MySQL\MySQL Server 5. 7\Data。可以通过配置向导或手工配置修改数据库的默认存放位置。

系统数据库

3.1.2　MySQL 自动建立的数据库

MySQL 安装完成之后，将会在其 data 目录下自动创建几个必需的数

据库，可以使用 show databases 命令来查看当前所有存在的系统数据库，如表 3-1 所示。

表 3-1　MySQL 系统数据库

数据库名称	数据库作用
mysql	描述用户访问权限
information_schema	保存关于 MySQL 服务器所维护的所有其他数据库的信息。如数据库名、数据库的表、表栏的数据类型与访问权限等
performance_schema	主要用于收集数据库服务器性能参数
sakila	MySQL 官方测试用的数据库
sys	sys 数据库里面包含了一系列的存储过程、自定义函数以及视图，存储了许多系统的元数据信息
world	存储当前世界上的主要城市、国家和语言信息

3.1.3　查看数据库

成功安装数据库后，可以使用 show databases 命令查看 MySQL 服务器中的所有数据库信息。

【例 3-1】 使用 show databases 语句查看 MySQL 服务器中的所有数据库。

命令和运行结果如下：

```
mysql > show databases;
        +--------------------+
        | Database           |
        +--------------------+
        | information_schema |
        | mysql              |
        | performance_schema |
        | sakila             |
        | sys                |
        | world              |
        +--------------------+
        6 rows in set (0.36 sec)
```

从例 3-1 的运行结果可以看出，通过 show databases 命令可以查看 MySQL 服务器中的所有数据库，结果显示 MySQL 服务器中的 6 个数据库。

3.2　MySQL 数据库的设计过程

数据库设计是指对于一个给定的应用环境，构造优化的数据库逻辑模式和物理结构，并据此建立数据库及其应用系统，使之能够有效地存储和管理数据，满足各种用户的应用需求，包括信息管理要求和数据操作要求。

数据库设计的目标是为用户和各种应用系统提供一个信息基础设施和高效率的运行环境。高效率的运行环境包括：数据库数据的存取效率、数据库存储空间的利用率、数据库系统运行管理的效率等。

以高校的教务管理系统为例，就需要数据库来存储学生的学籍信息、考试信息、教师信

息、课程信息等。数据库技术可以实现更加有效地管理和存取大量的数据资源，以提高人力、物力和财力的利用率和工作效率。

3.2.1 数据库设计的基本过程

一般来说，按照数据库规范化设计的方法，数据库设计可分为需求分析、概念设计、逻辑设计和物理设计 4 个阶段，如图 3-1 所示。之后是软件开发阶段中的数据库创建和数据库运行与维护。在实际的项目开发中，如果系统的数据关系较复杂，数据存储量较大，设计的表较多，表和表之间的关系比较复杂，就需要首先考虑规范的数据库设计，然后再进行具体的创建库、创建表的工作。数据库设计的步骤主要包括如下内容。

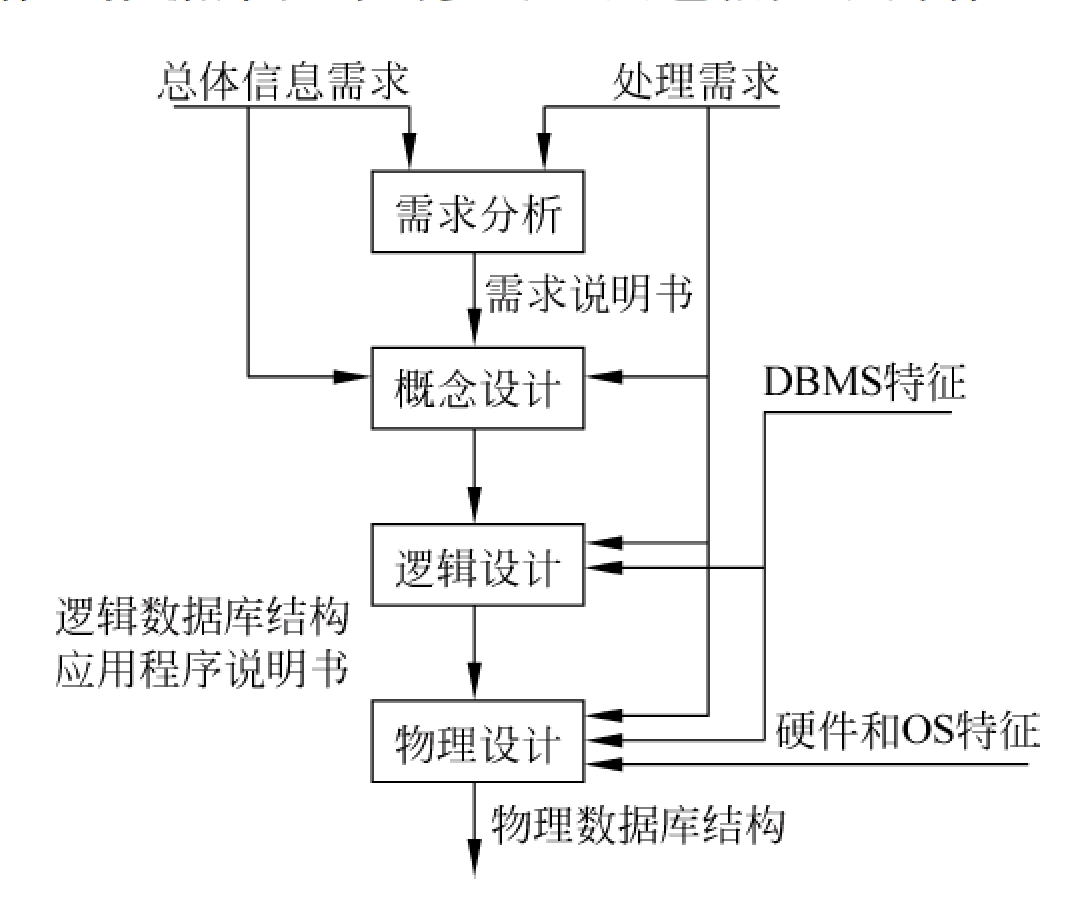


图 3-1 数据库设计的步骤

(1) 需求分析。需求分析的目标是通过调查研究，了解用户的数据要求和处理要求，并按一定的格式整理形成需求说明书。需求说明书是需求分析阶段的成果，也是以后设计的依据，它包括数据库所涉及的数据、数据的特征、数据量和使用频率的估计等。如数据名、属性及其类型、主关键字属性、保密要求、完整性约束条件、使用频率、更改要求、数据量估计等。

(2) 概念设计。概念设计是数据库设计的第 2 阶段，其目标是对需求说明书提供的所有数据和处理要求进行抽象与综合处理，按一定的方法构造反映用户环境的数据及其相互联系的概念模型。这种概念数据模型与 DBMS 无关，是面向现实世界的数据模型，极易为用户所理解。为保证所设计的概念数据模型能正确、完全地反映用户(单位)的数据及其相互关系，便于进行所要求的各种处理，在本阶段设计中可吸收用户参与和评议设计。

实体关系(E-R)的数据库设计方法是目前最常用的方法。基于实体关系的数据库设计方法的基本思想是在需求分析的基础上，用 E-R 图构造一个纯粹反映现实世界实体之间内在关系的企业模式，然后再将此企业模式转换成选定的 DBMS 上的概念模式。每个实体或联系将来就映射为一个数据表。

(3) 逻辑设计。逻辑设计阶段的设计目标是把上一阶段得到的与 DBMS 无关的概念数据模型转换成等价的，并为某个特定的 DBMS 所接受的逻辑模型所表示的概念模式，同时将概念设计阶段得到的应用视图转换成特定 DBMS 下的应用视图。在转换过程中要进一步落实需求说明，并满足 DBMS 的各种限制。逻辑设计阶段的结果是 DBMS 提供的数据

定义语言(DDL)写成的数据模式。

(4) 物理设计。物理设计阶段的任务是把逻辑设计阶段得到的逻辑数据库在物理上加以实现,其主要内容是根据 DBMS 提供的各种手段,设计数据的存储形式和存取路径,如文件结构、索引设计等,即设计数据库的内模式或存储模式。数据库的内模式对数据库的性能影响很大,应根据处理需求及 DBMS、操作系统和硬件的性能进行精心设计。

在数据库设计的基本过程中,每一阶段设计基本完成后,都要进行认真的检查,看看是否满足应用需求,是否符合前面已执行步骤的要求和满足后续步骤的需要,并分析设计结果的合理性。数据库设计完成后,就可以利用 MySQL 创建数据库了。

3.2.2 教务管理数据库设计的规范化

数据库应用程序的性质和复杂性可以使得数据库的设计过程变化很大。一个简单的数据库的设计,可以依赖于设计者的技巧和经验,采用直接设计数据库的方式进行。而对于为成千上万的客户处理事务的数据库,数据库设计可能是长达数百页的正式文档,其中需要包含有关数据库的各种可能细节。要进行较复杂的数据库设计,必须遵守数据库设计规范化规则(Normalization Rules),并按照软件工程提供的规范才能进行数据库设计。

按照规范化规则设计数据库,可以将数据冗余降至最低,使得应用程序软件可以在此数据库中轻松实现强制完整性,且很少包括执行涉及 4 个以上表的查询。规范化理论就是为了设计好的基本关系,使每个基本关系独立表示一个实体,并且尽量减少数据冗余。满足一定条件的关系模式称为范式(Normal Form,NF),一个低级范式的关系模式,通过分解(投影)方法可转换成多个高一级范式的关系模式的集合,这个过程称为规范化。

1. 按照规范化规则设计数据库

数据依赖是一个关系内部属性与属性之间的一种约束关系。这种约束关系是通过属性间值的相等与否体现出来的数据间的相关联系,它是现实世界属性间相互联系的抽象,是数据内在的性质,是语义的体现。人们提出了许多种类型的数据依赖,其中最重要的是函数依赖(Function Dependency,FD)和多值依赖(Multivalued Dependency,MVD)。而函数依赖极为普遍地存在于现实世界中。比如描述一个学生的关系 student,可以有学号、姓名、性别、电话等几个属性。由于一个学号只对应一个学生,所以一旦“学号”值确定后,学生的姓名、性别、电话等值也就被唯一地确定了。

例如:建立一个描述学校教务的数据库 teaching,该数据库涉及的对象包括学生学号、学生姓名、学生性别、电话、课程号、课程名称和成绩等数据项。假设用一个单一的关系模式学生来表示,则该关系模式的属性集合为:

```
U={学生学号,学生姓名,学生性别,电话,课程号,课程名称,成绩}
```

考察这个关系模式发现存在以下问题:

(1) 数据冗余度大:课程号和课程名称重复出现,重复次数与该班所有学生的所有课程成绩出现次数相同。

(2) 更新异常:由于数据冗余,当更新数据库中的数据时,系统要付出很大的代价来维护数据库的完整性,否则会面临数据不一致的危险。

(3) 插入异常:如果一门课程刚刚开设,尚无学生选课记录,则系统无法把该课程信息

存入数据库。

(4) 删除异常：如果某一级的学生全部毕业了，在删除该班学生信息的同时，把这个课程的信息也一起删除掉了。

鉴于存在以上种种问题，可以得出这样的结论：学生关系模式不是一个规范化的关系模式，一个规范化的关系应当不会发生插入异常、删除异常和更新异常，数据冗余度应尽可能地小。

2. 教务管理数据库的规范设计

为了避免上述诸多异常，在数据库设计时，需要遵守称为数据库范式的规则。下面以教务管理为例，介绍数据库设计中的范式(Normal Form，NF)理论。

(1) 第一范式(1NF)。第一范式的目标是确保每列的原子性。如果每列都是不可再分的最小数据单元，则满足第一范式(1NF)。

现以学生表为例，设计学生表结构如下：

```
学生(学生学号,学生姓名,学生性别,电话,课程号,课程名称,成绩)
```

以上学生表中各项都符合 1NF 条件。

(2) 第二范式(2NF)。第二范式是在第一范式的基础上，要求确保表中的每列都和码相关，即每一个非主属性都要完全函数依赖于码。

分析学生关系模式，码应该为(学生学号，课程号)，很明显，在该关系模式中，学生姓名、电话名称、学生性别、电话等只完全函数依赖于学生学号，因此对码(学生学号，课程号)是部分函数依赖，因此该关系模式不满足第二范式。

如果把学生相关的属性单独拿出来，形成关系模式：

```
学生(学生学号,学生姓名,学生性别,电话,课程号,课程名称)
选修(学生学号,课程号,成绩)
```

则以上两个关系模式都符合第二范式。

(3) 第三范式(3NF)。第三范式是在第二范式的基础上，要求确保表中的每列都和码直接相关，而不是间接相关。如果一个关系满足 2NF，并且除了码以外的其他列都不相互依赖，则满足第三范式(3NF)。

为了理解第三范式，需要根据 Armstrong 公理之一定义传递函数依赖，假设 A、B 和 C 是关系模式 R 的三个属性，如果 A→B 且 B→C，则从这些函数依赖中，可以得出 A→C。

考察上述分解后的关系模式：

```
学生(学生学号,学生姓名,学生性别,电话,课程号,课程名称)
```

可以得出课程名称→课程号，而课程号→学生学号(假设学生选修该课程)，因此存在课程名称→学生学号的传递函数依赖，因此该关系模式不符合第三范式。

如果把学生关系模式中的课程相关的属性单独拿出来，形成关系模式：

```
学生(学生学号,学生姓名,学生性别,电话)
课程(课程编号,课程名称)
```

则以上两个关系模式都满足第三范式。

分析学生实体和课程实体之间的联系，并添加一些必要的属性，可以得出学生实体和课

程实体之间是多对多(m∶n)的联系,因此,绘制学生实体和课程实体的“选修”关系局部E-R图,如图3-2所示。如果再加上教师实体,并针对本系统的特点修改,则教务管理系统的E-R图如图3-3所示。

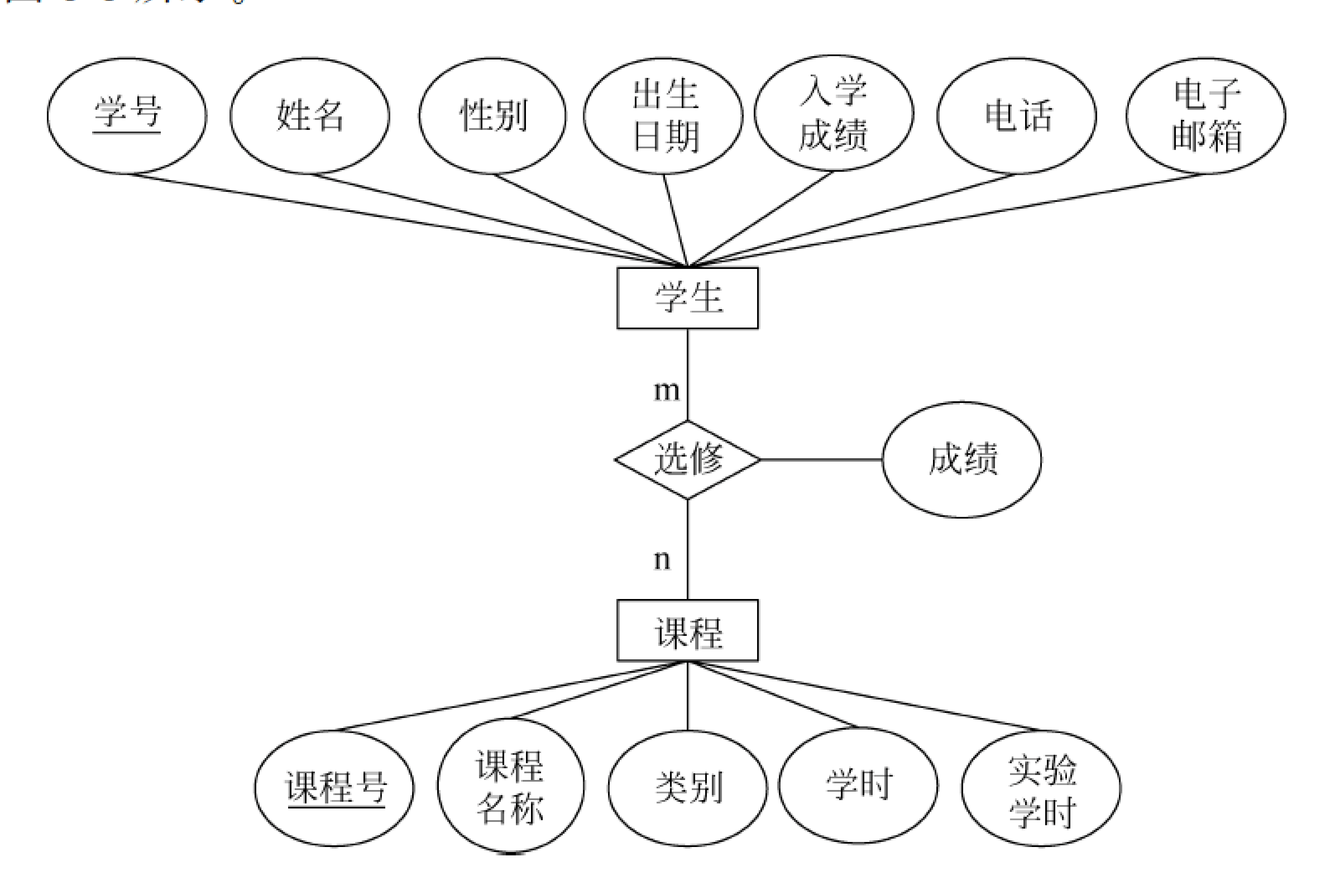


图3-2　学生-课程“选修”关系E-R图

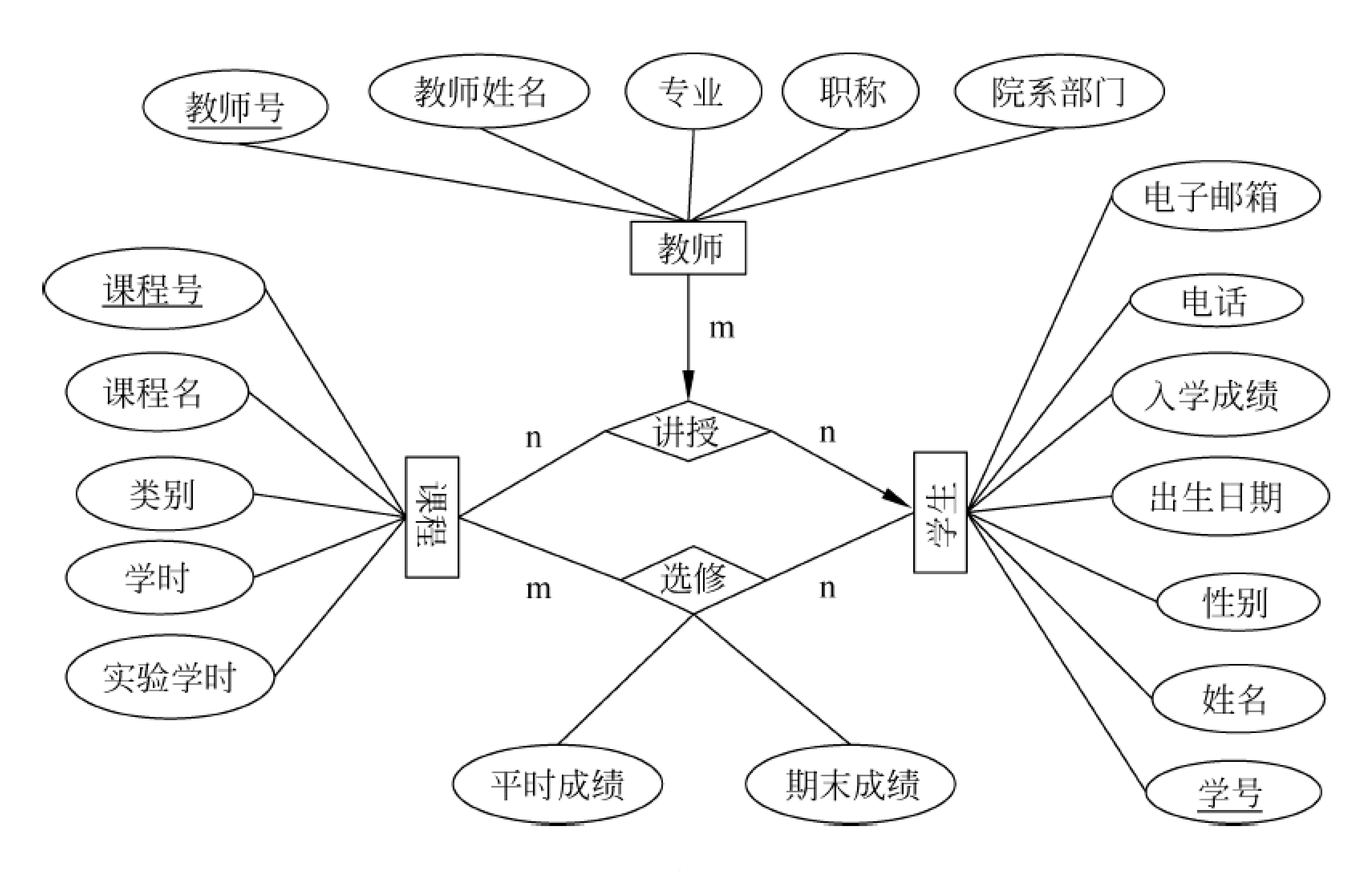


图3-3　教务管理系统E-R图

之后,就可以在此基础上根据E-R图的转换,利用MySQL等软件创建teaching数据库和student、score、course、teacher等数据表了。

3.3　用户数据库的创建和管理

本节利用MySQL语句创建和维护教务管理数据库,用于存储学校的学生、课程、教师和成绩等基本数据。本节建立的名称为teaching的教务数据库,将在本书中作为数据库管理的示例数据库应用。

3.3.1 创建数据库

在创建数据库时，数据库命名有以下几项规则：

- 不能与其他数据库重名，否则将发生错误。
- 名称可以由任意字母、阿拉伯数字、下画线(_)和“$”组成，可以使用上述的任意字符开头，但不能使用单独的数字，否则会造成它与数值相混淆。
- 名称最长可为 64 个字符，而别名最多可长达 256 个字符。
- 不能使用 MySQL 关键字作为数据库名、表名。

在默认情况下，Windows 下数据库名、表名的大小写是不敏感的，而在 Linux 下数据库名、表名的大小写是敏感的。如果为了便于数据库在平台间进行移植，可以采用小写来定义数据库名和表名。

1. 创建数据库语法结构

使用 create database 或 create schema 命令可以创建数据库，其语法结构如下：



创建数据库

```
create {database|schema}[if not exists]databasename
[default]character set charset_name
|[default]collate collation_name;
```

说明：

(1) create database|schema：创建数据库的命令。MySQL 中 schema 也是指数据库。

(2) if not exists：如果已存在某个数据库，再来创建一个同名的库，这时会出现错误信息。为避免错误信息，可以在建库前加上这一判断，只有该库目前尚不存在时才执行 create database 操作。

(3) databasename：数据库标识符名。

(4) [default] character set charset_name：default character set 指定数据库的默认字符集(Charset)，charset_name 为字符集名称。创建数据库时最好指定字符集。

(5) [default] collate collation_name：collate 指定字符集的校对规则，collation_name 为校对规则名称。

2. 创建数据库

创建数据库是指在数据库系统中划分一块空间，用来存储相应的数据。这是进行表操作的基础，也是进行数据库管理的基础。MySQL 中，创建数据库是通过 SQL 语句 create database 实现的。

【例 3-2】 通过 create database 语句创建一个名称为 mysqltest 的数据库。

命令和运行结果如下：

```
mysql > create database if not exists mysqltest;
       Query OK, 1 row affected (0.05 sec)
```

结果表明，创建 mysqltest 数据库成功。

【例 3-3】 创建教务管理数据库 teaching，并指定字符集为 gb 2312，校对原则为 gb2312_chinese_ci。

命令和运行结果如下：

```
mysql > create database teaching
    -> default character set gb2312
    -> default collate gb2312_chinese_ci;
       Query OK, 1 row affected (0.00 sec)
```

数据库 teaching 创建后，默认的字符集为 gb 2312，校对原则为 gb2312_chinese_ci，即识别简体中文，且字母不区分大小写。

3. 查看新创建的数据库

成功创建数据库后，可以使用 show databases 命令查看数据库，也可以在指定路径（或数据库的默认存放位置）下查看数据库。例如查看 teaching，如图 3-4 所示。



图 3-4　查看已创建数据库

3.3.2　管理数据库



管理数据库

1. 打开数据库

数据库创建后，若要操作一个数据库，还需要使其成为当前的数据库，即打开数据库。可以使用 use 语句打开一个数据库，使其成为当前默认数据库。

例如，选择名称为 mysqltest 的数据库，设置其为当前默认的数据库。

命令和运行结果如下：

```
mysql > use mysqltest;
       Database changed
```

其中，Database changed 表明数据库 mysqltest 已经打开，变成当前数据库了，可以在数据库 mysqltest 中进行相关的操作了。

2. 修改数据库

数据库创建后，如果需要，可以修改数据库的参数。

修改数据库的语法格式如下：

```
alter {database | schema} [db_name]
[default] character set charset_name
|[default] collate collation_name;
```

其中，alter 是修改数据库的命令关键字。

【例 3-4】 将 mysqltest 库修改字符集为 gb 2312，校对原则为 gb2312_chinese_ci。

命令和运行结果如下：

```
mysql > alter database mysqltest
    - > default character set gb2312
    - > collate gb2312_chinese_ci;
       Query OK, 1 row affected (0.00 sec)
```

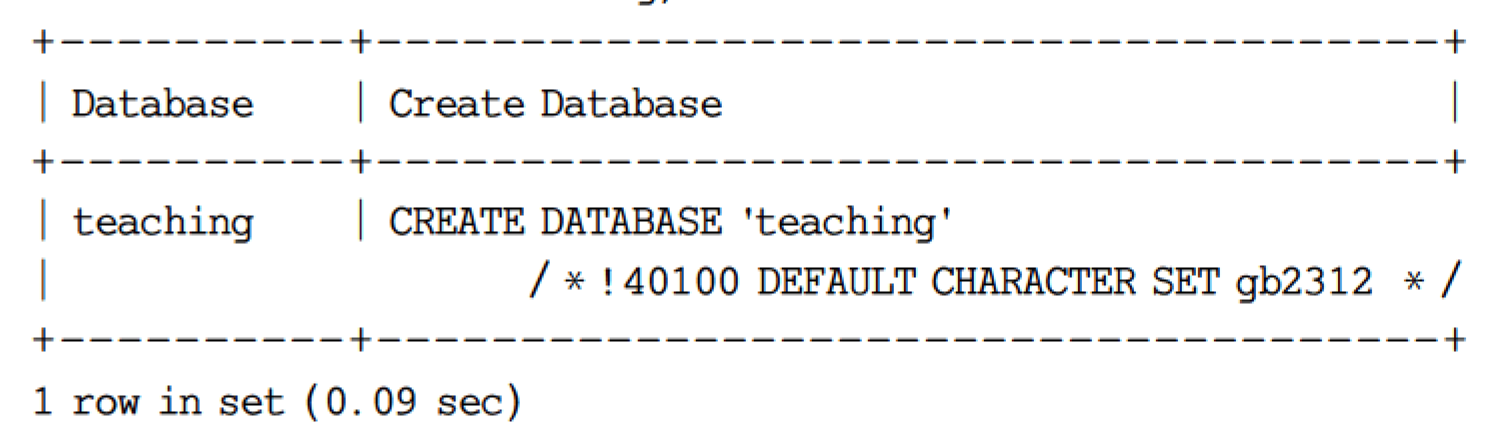
3. 显示数据库结构

如果查看一数据库的相关信息，例如 MySQL 版本 id 号、默认字符集等信息，使用 MySQL 命令实现。

【例 3-5】 显示数据库 teaching 的结构信息。

命令和运行结果如下：

```
mysql > show create database teaching;
```



```
+-----------+---------------------------------------+
| Database  | Create Database                       |
+-----------+---------------------------------------+
| teaching  | CREATE DATABASE 'teaching'
|                    / * !40100 DEFAULT CHARACTER SET gb2312  * /
+-----------+---------------------------------------+
1 row in set (0.09 sec)
```

4. 删除数据库

删除数据库是指在数据库系统中删除已经存在的数据库。删除数据库之后，原来分配的空间将被收回。删除数据库的语法格式如下：

```
drop database [if exists] db_name
```

例如，删除 mysqltest 库命令如下：

```
mysql > drop database mysqltest;
```

需要注意的是，删除数据库会删除该数据库中所有的表和所有数据。因此，删除数据库前最好存有备份。

3.4 利用 MySQL Workbench 管理数据库

在图形管理工具 MySQL Workbench 窗口中，是通过使用可视化的界面的提示来创建数据库和维护数据库的。

3.4.1 利用 MySQL Workbench 创建数据库



利用 Workbench 创建数据库

利用 MySQL Workbench 创建数据库的步骤如下：

(1) 从“开始”→“所有程序”中找到 MySQL 文件夹，执行 MySQL Workbench 6.2 CE 命令，进入 MySQL Workbench 主界面，单击如图 3-5 所示的 MySQL57 连接。进入 MySQL Workbench 数据库操作的主界面，如图 3-6 所示。

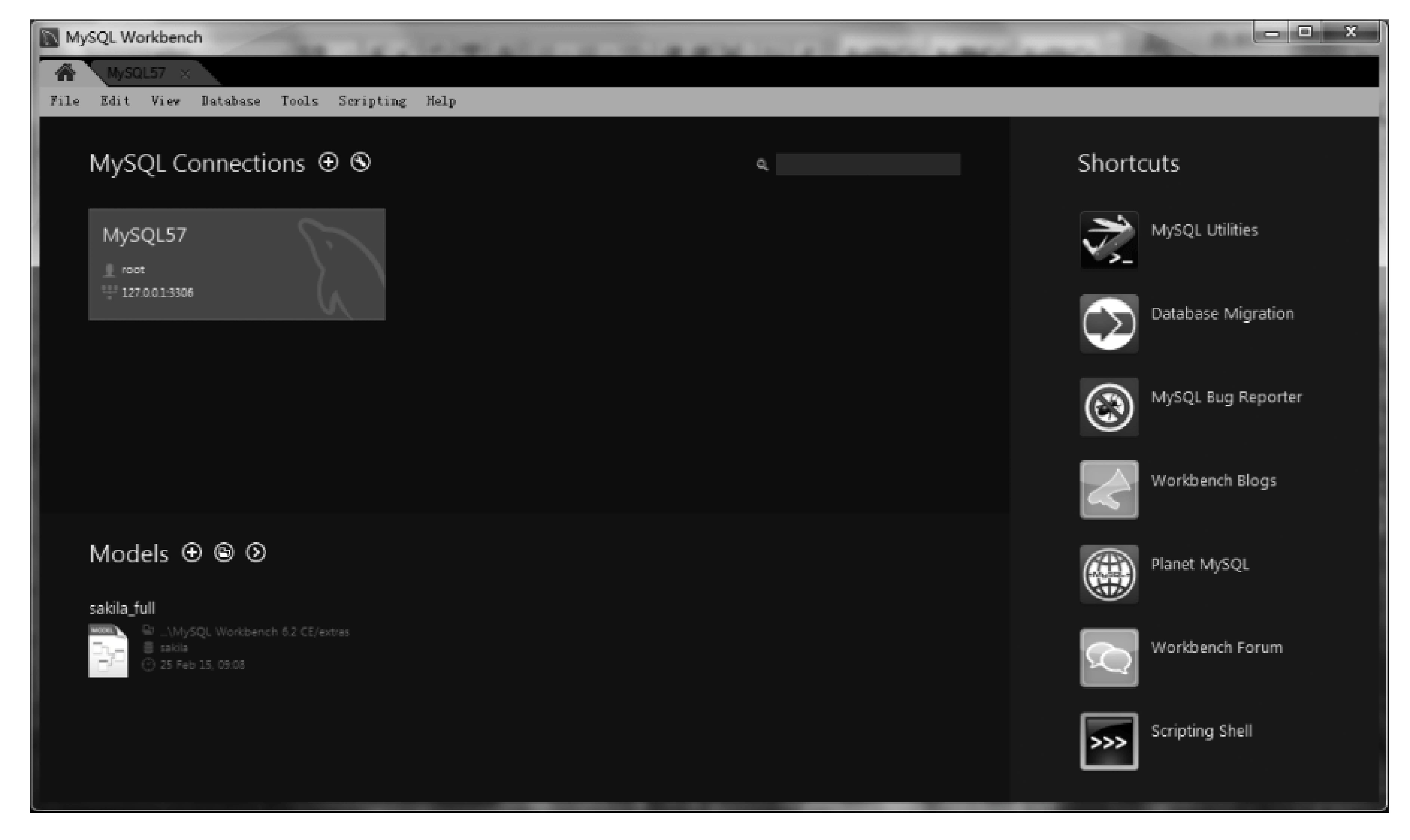


图 3-5 MySQL Workbench 主界面

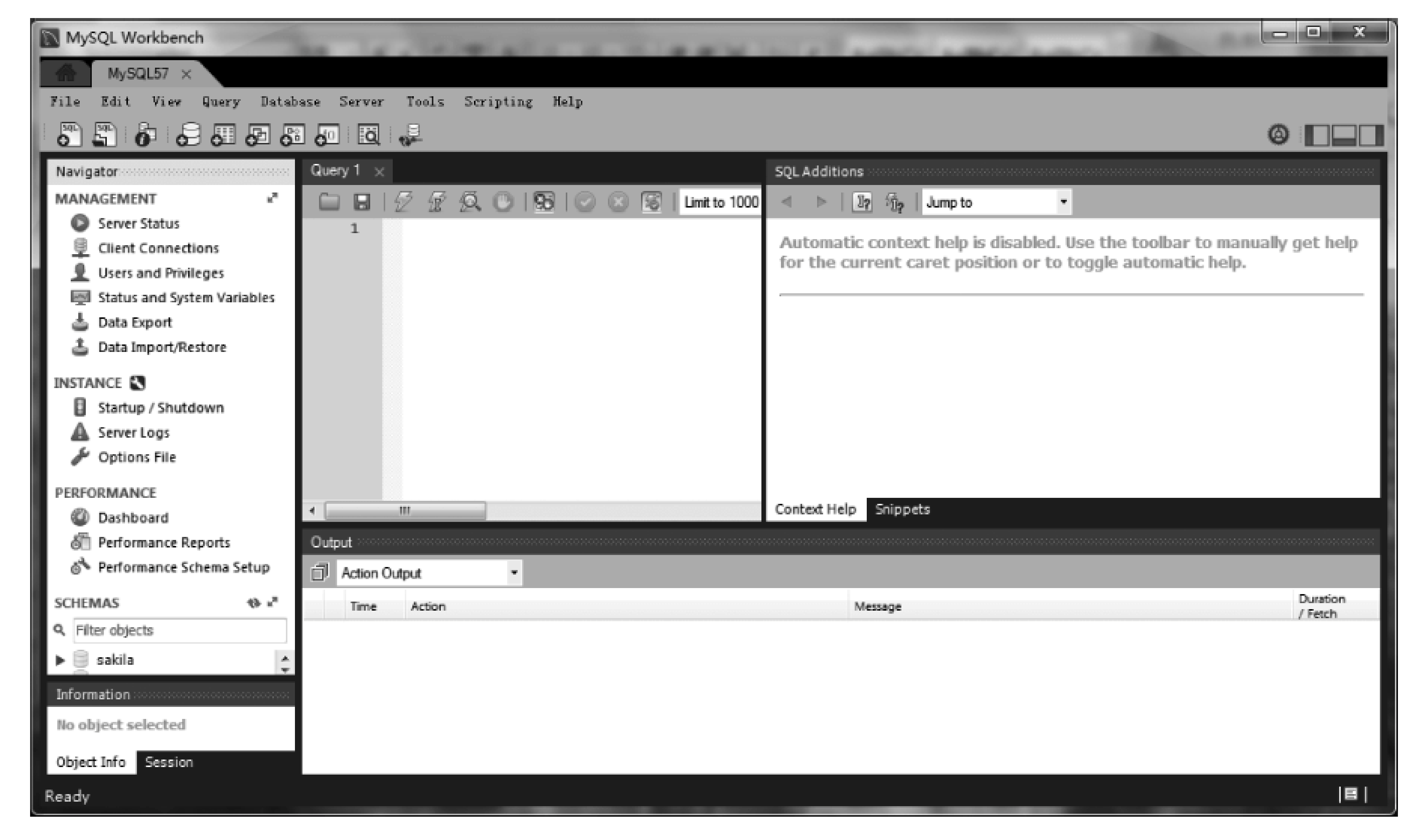


图 3-6 数据库操作主界面

（2）在数据库操作主界面中，单击如图 3-7 所示的提示为 Create a new schema in the connected server 的创建数据库的按钮，schema 在这里就是数据库的意思。

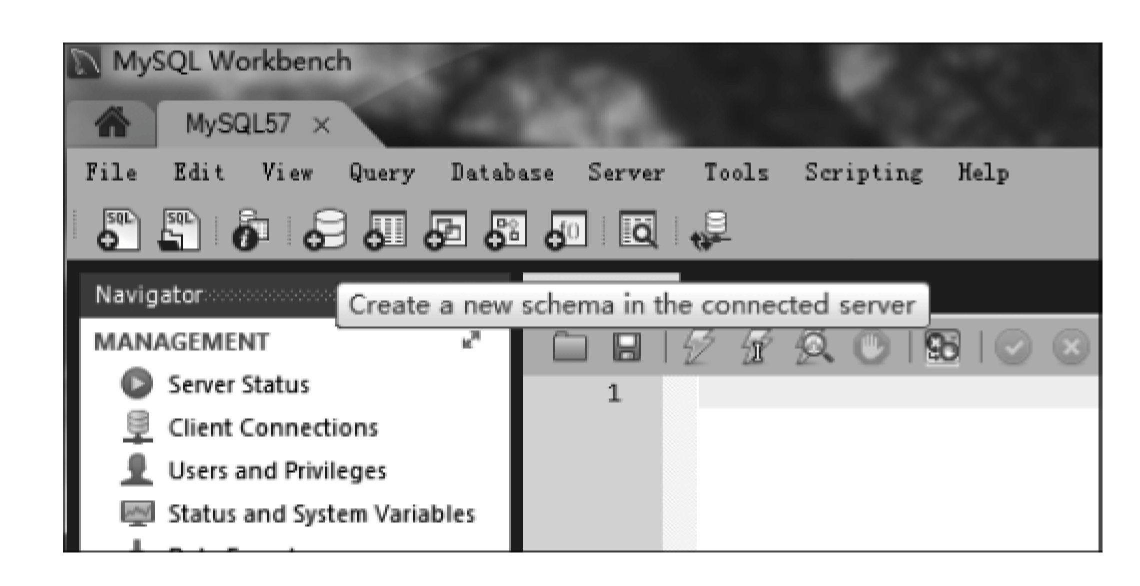


图 3-7 数据库创建操作

（3）进入如图 3-8 所示的设置数据库参数的界面，输入数据库名 MySQLtest，选择字符集和排序规则，然后单击 Apply 按钮。

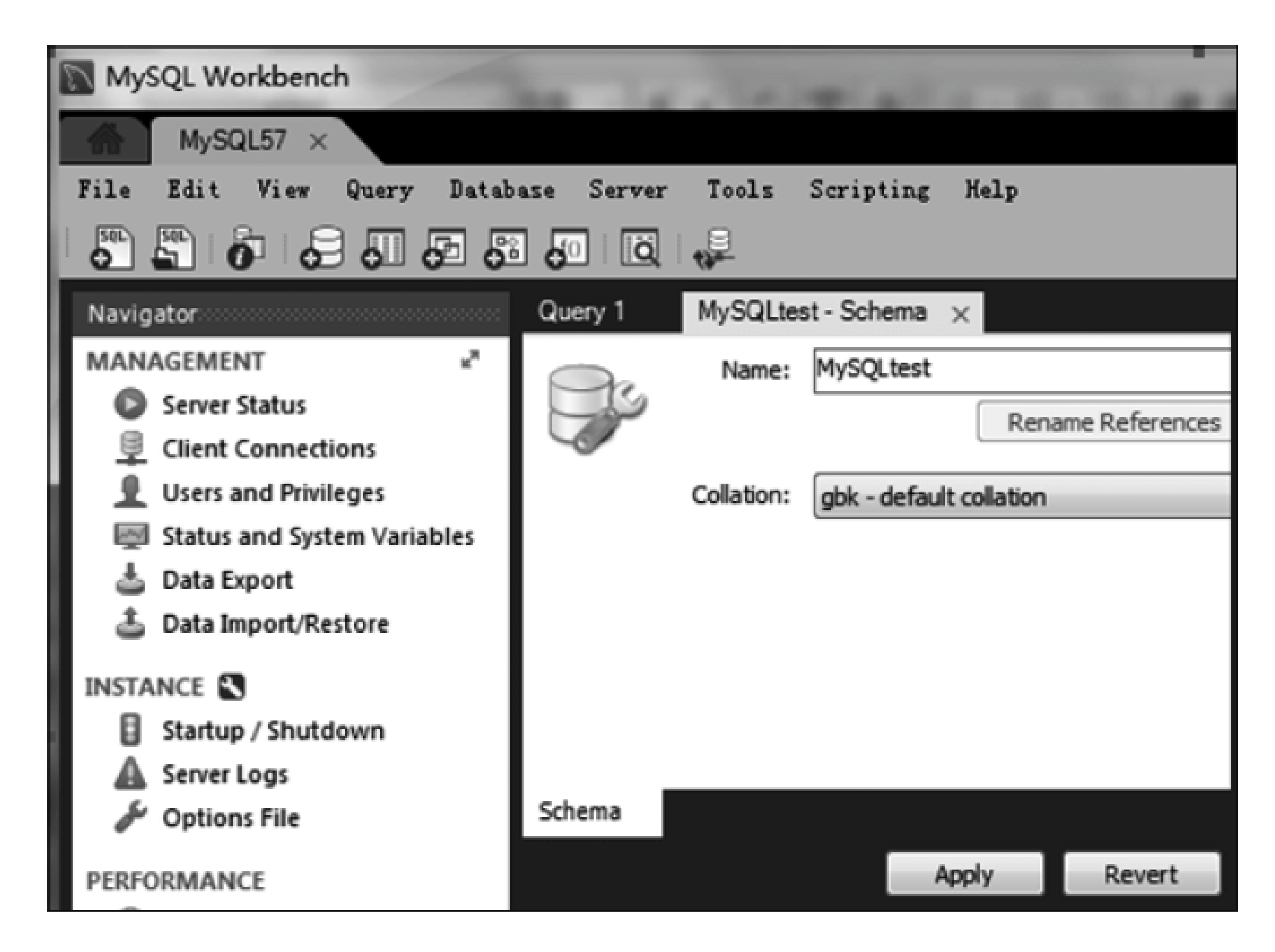


图 3-8　数据库参数选择

（4）进入如图 3-9 所示的创建数据库脚本显示的对话框中，再单击 Apply 按钮，即可完成数据库的创建。

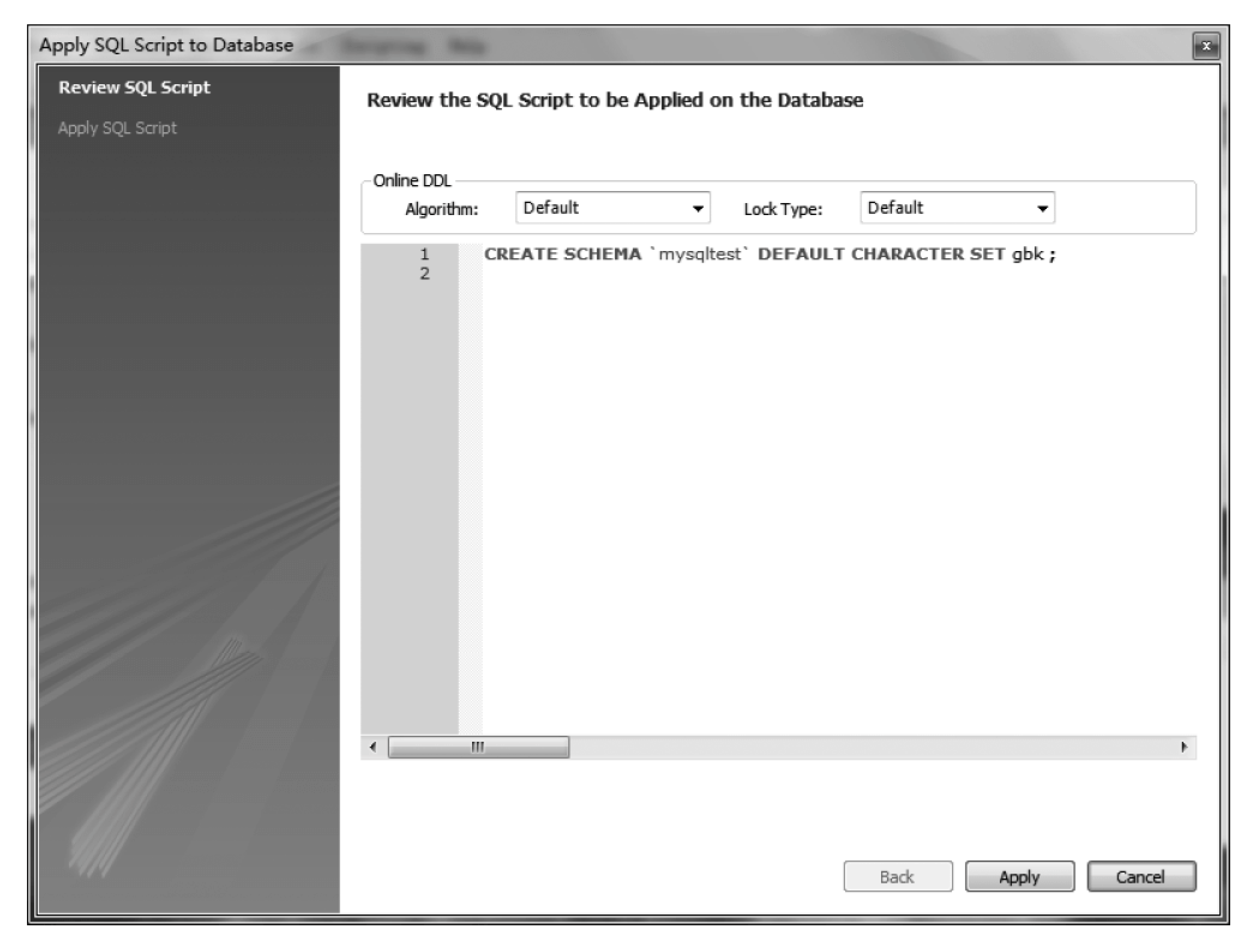
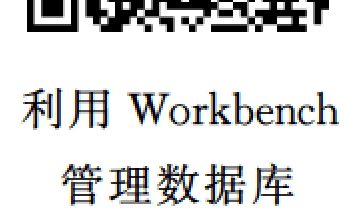


图 3-9　数据库脚本显示

(5) 查看数据库。在数据库操作界面的 schemas 区域，就可以看到刚才创建的数据库 mysqltest，还有前面创建的数据库 teaching 以及 MySQL 中的系统数据库了，如图 3-10 所示。

3.4.2 利用 MySQL Workbench 管理数据库



利用 Workbench 管理数据库

1. 修改数据库参数

(1) 在数据库操作界面的 SCHEMAS 区域，右击要修改的数据库 mysqltest，如图 3-11 所示。执行弹出菜单中的 Alter Schema 命令。

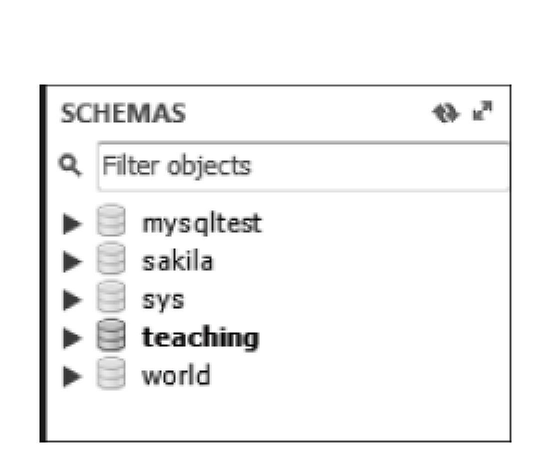


图 3-10 查看数据库



图 3-11 修改数据库命令

(2) 进入如图 3-12 所示的修改数据库对话框，可以对数据库 mysqltest 进行修改。最后单击 Apply 按钮即可完成数据库的修改。

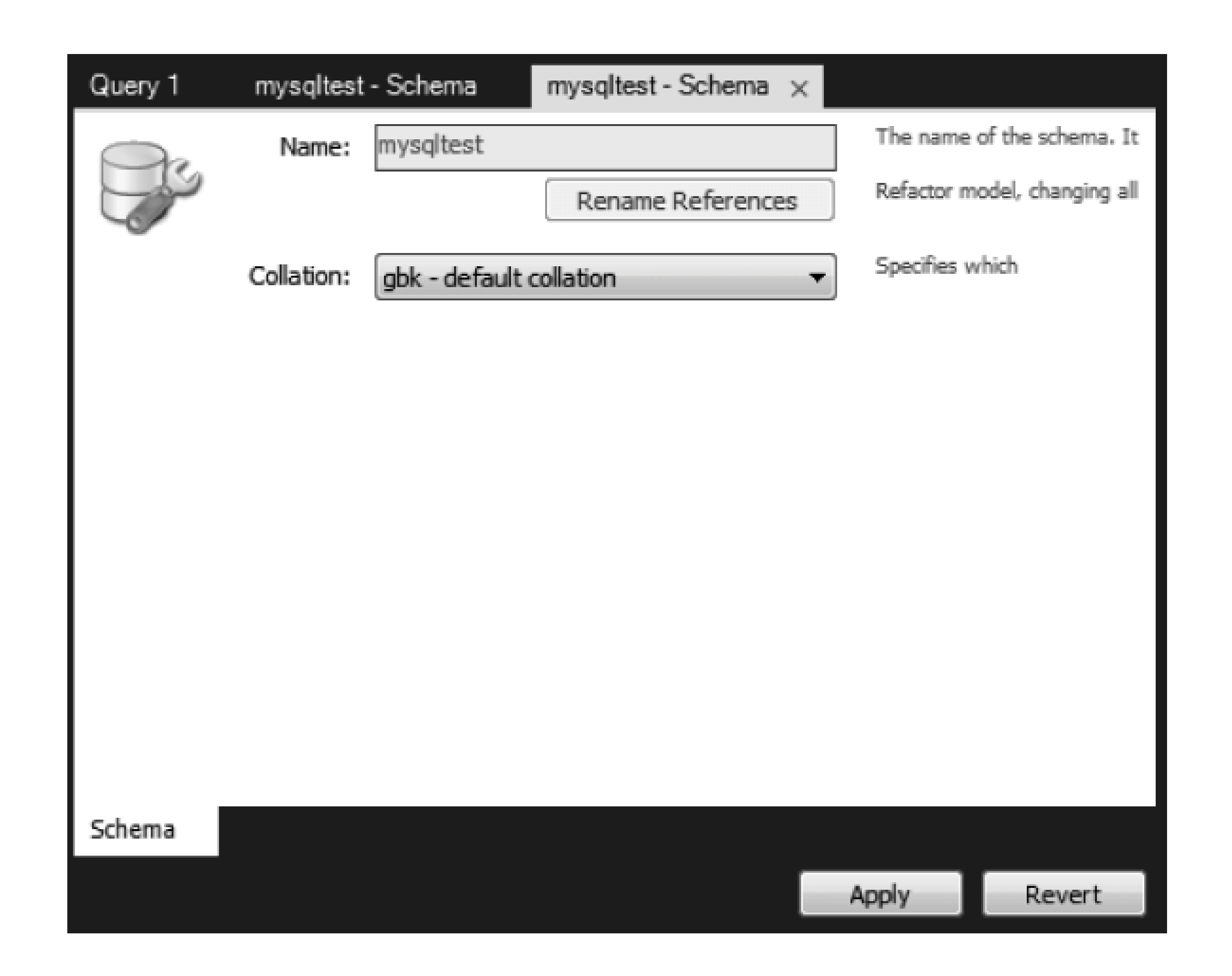


图 3-12 修改数据库

2. 删除数据库

(1) 在数据库操作界面的 SCHEMAS 区域，右击要修改的数据库 mysqltest，如图 3-13 所示。执行弹出菜单中的 Drop Schema 命令。进入删除数据库对话框，如图 3-14 所示。

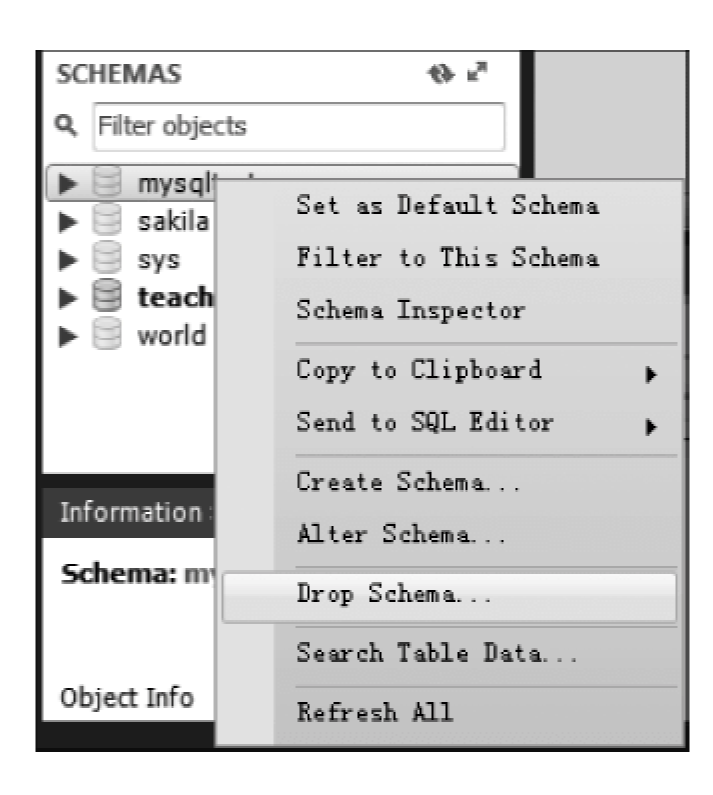



图 3-13　删除数据库命令

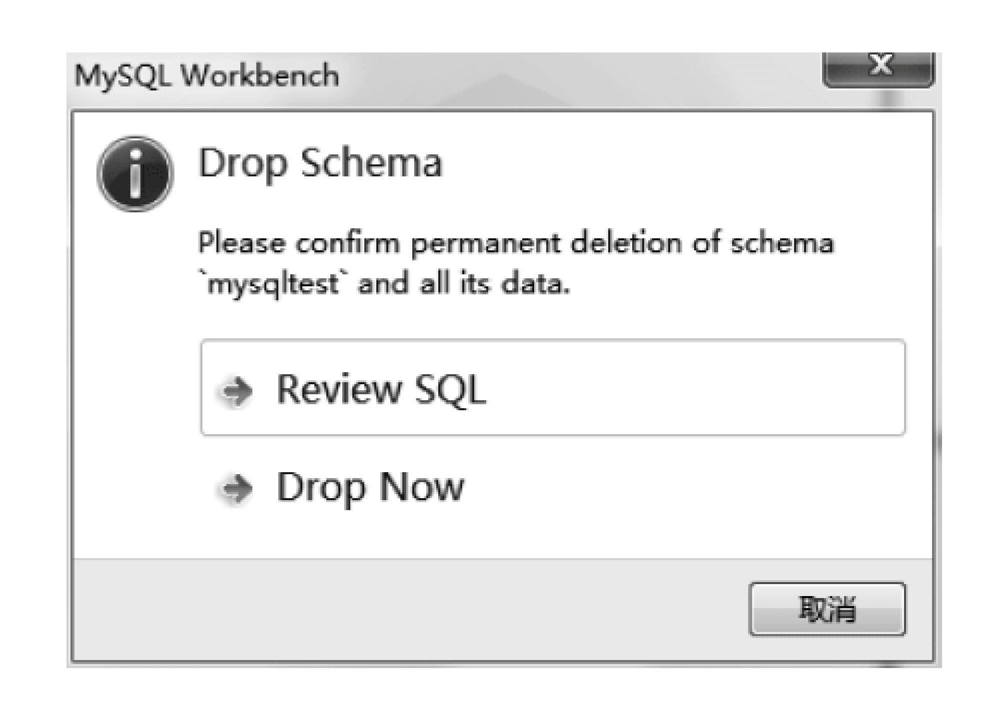


图 3-14　复审删除操作

(2) 选择 Review SQL 选项，进入如图 3-15 所示的复审对话框。单击 Cancel 按钮，即可取消数据库删除的过程，单击 Execute 按钮，即可删除该数据库。

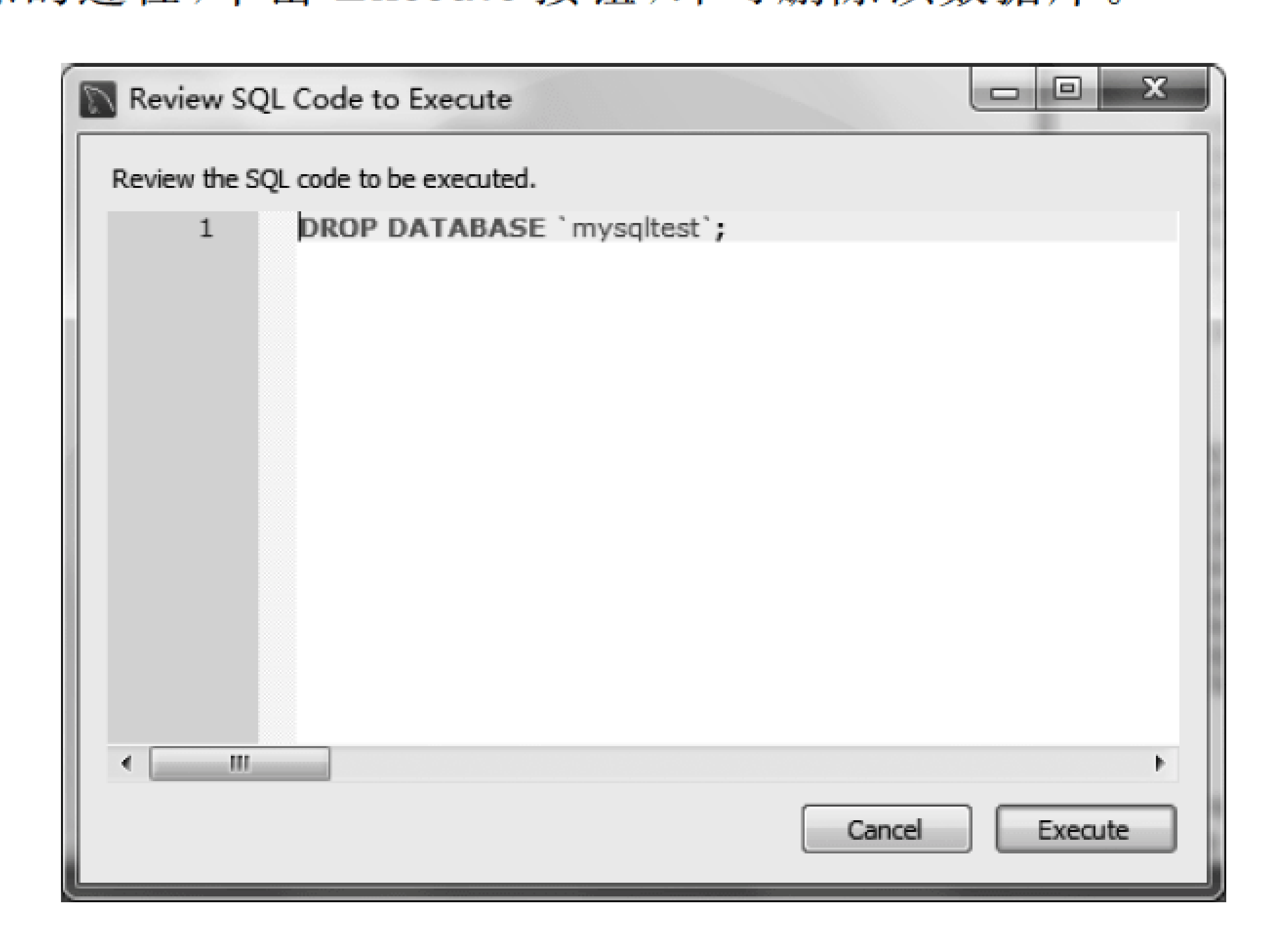


图 3-15　复审对话框

利用图形管理工具 MySQL Workbench 管理数据库，是对 MySQL 数据库进行可视化操作的一种方式，比较适合初学者使用。

3.5　MySQL 存储引擎

在 Oracle 和 SQL Server 等数据库中只有一种存储引擎，所有数据存储管理机制都是一样的。而 MySQL 数据库提供了多种存储引擎，用户可以根据不同的需求为数据表选择不同的存储引擎，用户也可以根据自己的需要编写自己的存储引擎，MySQL 的核心就是存储引擎。

数据库的存储引擎决定了表在计算机中的存储方式。存储引擎就是如何存储数据、如何为存储的数据建立索引和如何更新、查询数据等技术的实现方法。因为在关系数据库中数据的存储是以表的形式存储的，所以存储引擎简而言之就是指表的类型。

MySQL 5.7 支持的存储引擎有 InnoDB、MRG_MyISAM、Memory、BLACKHOLE、

MyISAM、CSV、Archive、Federated 和 PERFORMANCE schema 9 种。不同的存储引擎都有各自的特点，以适应不同的需求。MySQL 常用存储引擎如表 3-2 所示。

表 3-2　MySQL 常用存储引擎功能对比

功　能	InnoDB	MyISAM	Memory
存储限制	64TB	256TB	RAM
支持事务	支持	无	无
空间使用	高	低	低
内存使用	高	低	高
支持数据缓存	支持	无	无
插入数据速度	低	高	高
支持外键	支持	无	无

3.5.1　查看数据库存储引擎

MySQL 的存储引擎是一种插入式的存储引擎概念。这决定了 MySQL 数据库中的表可以用不同的方式存储。用户可以根据自己的不同要求，选择不同的存储方式、是否进行事务处理等。MySQL 的默认存储引擎是 InnoDB，如果想设置其他存储引擎，可以使用如下 MySQL 命令：

```
set default_storage_engine = MyISAM;
```

该命令可以临时将 MySQL 当前会话的存储引擎设置为 MyISAM，使用 MySQL 命令“show engines;”可以查看当前 MySQL 服务实例默认的存储引擎。

例如，执行“show engines;”，即可查看当前 MySQL 服务实例支持的存储引擎，命令和结果如图 3-16 所示。

```
mysql> SHOW  ENGINES ;
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO         |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |
| FEDERATED          | NO      | Federated MySQL storage engine                                 | NULL         | NULL | NULL       |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
9 rows in set (0.04 sec)
```

图 3-16　MySQL 数据库支持的存储引擎类型

3.5.2　常用存储引擎介绍

1. 存储引擎 InnoDB

MySQL 5.7 选择 InnoDB 作为默认存储引擎。InnoDB 是事务型数据库的首选引擎，是具有提交、回滚和崩溃恢复能力的事务安全存储引擎，支持行锁定和外键约束。

InnoDB 是 MySQL 上第一个提供外键约束的表引擎，而且 InnoDB 对事务处理的能力，也是 MySQL 其他存储引擎所无法与之比拟的。相对 MEMORY 来说，写处理能力差些，且会占用较多磁盘空间以保留数据和索引。

InnoDB 存储引擎中支持自动增长列 auto_increment。自动增长列的值不能为空，且值必须唯一。MySQL 中规定自增列必须为主键。在插入值时，如果自动增长列不输入值，则插入的值为自动增长后的值。如果输入的值为 0 或者空（null），则插入的值也为自动增长后的值。如果插入某个确定的值，且该值在前面没有出现过，则可以直接插入。

InnoDB 存储引擎除了支持外键（Foreign Key）和事务（Transaction）外还支持全文检索。因此，InnoDB 存储引擎能够提供 OLTP 支持，如果表需要执行大量的添加、删除、修改数据的操作，出于事务安全方面的考虑，InnoDB 存储引擎是更好的选择。

2. 存储引擎 MyISAM

MyISAM 存储引擎曾是 MySQL 的默认存储引擎。MyISAM 存储引擎不支持事务、外键约束，但访问速度快，对事务完整性不要求，适合于以 SELECT/INSERT 为主的表。

现在的 MyISAM 增加了很多有用的扩展。MyISAM 存储引擎的表存储成三个文件。文件的名字与表名相同。扩展名包括 frm、myd 和 myi。其中，frm 为扩展名的文件存储表的结构；myd 为扩展名的文件存储数据，其是 mydata 的缩写；myi 为扩展名的文件存储索引，其是 myindex 的缩写。

MyISAM 存储引擎的特点如下：

- 具有检查和修复表的大多数工具。
- MyISAM 表可以被压缩。
- MyISAM 表最早支持全文索引。
- MyISAM 表不支持事务。
- MyISAM 表不支持外键。

如果需要执行大量的 select 语句，出于性能方面的考虑，MyISAM 存储引擎是更好的选择。

3. 存储引擎 MEMORY

MEMORY 存储引擎是 MySQL 中的一类特殊的存储引擎。其使用存储在内存中的内容来创建表，而且所有数据也放在内存中。这些特性都与 InnoDB 存储引擎、MyISAM 存储引擎不同。每个 MEMORY 表可以放置数据量的大小受 max_heap_table_size 系统变量的约束，初始值为 16MB，可按需求增大。此外，在定义 MEMORY 表时可通过 max_rows 子句定义表的最大行数。

每个基于 MEMORY 存储引擎的表实际对应一个磁盘文件。该文件的文件名与表名相同，类型为 frm 类型。该文件中只存储表的结构。而其数据文件，都是存储在内存中的。这样有利于对数据的快速处理，提高整个表的处理效率。

该存储引擎主要用于那些内容稳定的表，或者作为统计操作的中间表。对于该类表需要注意的是，因为数据并没有实际写入磁盘，一旦重启，则会丢失。

值得注意的是，服务器需要有足够的内存来维持 MEMORY 存储引擎的表的使用。如果不需要使用了，可以释放这些内存，甚至可以删除不需要的表。

3.5.3 如何选择存储引擎

对于存储引擎的选择，在实际工作中，选择一个合适的存储引擎是一个很复杂的问题。每种存储引擎都有各自的优势，可以根据各种存储引擎的特点进行对比，给出不同情况下选

择存储引擎的建议。

MySQL 中提到了存储引擎的概念，它是 MySQL 的一个特性，可简单理解为后面要介绍的表类型。每一个表都有一个存储引擎，可在创建时指定，也可以使用 alter table 语句修改，都是通过 engine 关键字设置的。

3.6 小　结

本章主要介绍了创建数据库、删除数据库、MySQL 存储引擎的知识。创建和删除数据库是本章的重点。存储引擎的知识比较难理解，只要了解相应的知识即可，但一定要了解自己的 MySQL 5.7 数据库默认使用的存储引擎是 InnoDB。具体要求如下：

- 熟悉 MySQL 常用图形管理工具 Workbench 的功能及使用。
- 掌握 MySQL 数据库的创建方法。
- 掌握 MySQL 数据库的删除。
- 熟悉常见的存储引擎 InnoDB 的特点。

习　题　3

1. 选择题

(1) 在 MySQL 数据库中，通常使用________语句来指定一个已有数据库作为当前工作数据库。

A. using　　B. used　　C. uses　　D. use

(2) ________命令用于删除一个数据库。

A. create database　　B. drop database

C. alter database　　D. use InnoDB

(3) ________存储引擎支持外键(Foreign Key)和事务(Transaction)以及全文检索。

A. MEMORY　　B. MyISAM　　C. InnoDB　　D. MySQL

(4) 在创建数据库时，可以使用________子句确保如果数据库不存在就创建它，如果存在就直接使用它。

A. if not exists　　B. if exists　　C. if not exist　　D. if exist

(5) 在图形管理工具 MySQL Workbench 窗口中使用可视化的界面不能够支持________。

A. 数据库引擎开发　　B. 创建数据库　　C. 维护数据库　　D. 数据库设计

(6) MySQL 自带数据库中，________存储了系统的权限信息。

A. information_schema　　B. mysql

C. sakila　　D. performance_schema

2. 简答题

(1) 如何在 MySQL Workbench 窗口中修改数据库的字符集？

(2) 简述在 MySQL Workbench 窗口中创建数据库的步骤。

(3) 简述 MySQL 数据库的特点。

(4) 存储引擎 InnoDB、MyISAM 和 MEMORY 各有什么优缺点?

3. 上机练习题

题目要求:登录数据库系统以后,分别创建 student 和 teacher 数据库,数据库都创建成功后,删除 teacher 数据库。然后查看数据库系统中还存在哪些数据库。

主要实现过程如下所示:

(1) 登录 MySQL 数据库客户端;

(2) 查看数据库系统中已存在的数据库;

(3) 查看该数据库系统支持的存储引擎的类型;

(4) 创建 student 数据库和 teacher 数据库;

(5) 再次查看数据库系统中已经存在的数据库,确保 student 和 teacher 数据库已经存在;

(6) 删除 teacher 数据库;

(7) 再次查看数据库系统中已经存在的数据库,确保 teacher 数据库已经删除。

第 4 章 表及数据完整性

创建数据库的目的是为了存储、管理和查询数据，而表是存储数据的最重要的载体。表是 MySQL 数据库中最重要的数据库对象，也是构建高性能数据库的基础。数据表设计的优劣将影响磁盘空间的使用效率、数据处理时内存的利用率以及数据的查询效率。在这个过程中，要注意表结构的规范化、数据类型的正确选择，以及数据库和数据表字符集的统一问题。表的数据完整性规则是保证表中数据的正确性、精确性和可靠性的关键。

本章将通过建立一个教务管理数据库中的学生、课程、教师和成绩等数据表，介绍各种数据表的创建、修改、管理、存储与数据格式转换，以及实现数据完整性的方法和基本操作。

4.1 MySQL 数据库表的管理

在 MySQL 数据库系统中，可以按照不同的标准对表进行分类。

1. 按照表的用途分类

(1) 系统表：用于维护 MySQL 服务器和数据库正常工作的数据表。例如，系统数据库 MySQL 中就存在若干系统表。

(2) 用户表：由用户自己创建的、用于各种数据库应用系统开发的表。

(3) 分区表：分区表是将数据水平划分为多个单元的表，这些单元可以分布到数据库中的多个文件组中。在维护整个集合的完整性时，使用分区可以快速而有效地访问或管理数据子集，从而使大型表或索引更易于管理。

2. 按照表的存储时间分类

(1) 永久表：包括 SQL Server 的系统表和用户数据库中创建的数据表，该类表除非人工删除，否则一直存储在介质中。

(2) 临时表：临时表只有创建该表的用户在用来创建该表的连接中可见。临时表关联的连接被关闭时，临时表自动地被删除。如果服务器关闭，则所有临时表会被清空、关闭。

4.1.1 InnoDB 存储引擎的表空间

MySQL 5.7 的数据库表默认使用 InnoDB 存储引擎，InnoDB 表空间是 MySQL 数据表的存储空间的管理模式。在这种管理模式下，数据库中不但存储数据文件和重做日志文件，还有管理这些文件的表空间文件。InnoDB 表空间分为共享表空间和独享表空间两种类型。

1. 表空间的基本概念

(1) 共享表空间。MySQL 服务实例承载着数据库的所有 InnoDB 表的数据、索引、各种元数据以及事务回滚(undo)信息,全部存放在共享表空间文件中。默认情况下该文件位于数据库根目录下,文件名是 ibdata1,且文件的初始大小为 10MB。即 InnoDB 的所有文件共享一个表空间,其最大容量限制约为 64TB。

(2) 独立表空间。每一个表都将会以独立的文件方式来进行存储,每一个表都有一个.frm 表描述文件,还有一个.ibd 文件。该文件包括一个表单独的数据内容以及索引内容。

2. 查看数据库的表空间

利用如下命令可以查看数据库的表空间。

```
mysql > show variables like 'InnoDB_data % ';
```

表空间由 4 个文件组成:ibdata1、ibdata2、ibdata3、ibdata4,每个文件的大小为 10MB,当每个文件都满了的时候,ibdata4 会自动扩展。

不管是共享表空间还是独立表空间,都会存在 InnoDB_data_file 文件,因为这些文件不仅仅要存放数据,而且还要存储事务回滚(undo)信息。

3. 共享表空间和独立表空间的比较

(1) 共享表空间的特点。

- 表空间可以分成多个文件存放在一起方便管理。
- 多个表及索引在表空间中混合存储,当数据量非常大的时候,表做了大量删除操作后表空间中将会有大量的空隙,特别是对于统计分析,对于经常删除操作的这类应用最不适合用共享表空间。
- 共享表空间分配后不能回缩。

(2) 独立表空间的特点。

- 每个表都有独立的表空间。每个表的数据和索引都会存在自己的表空间中,可以实现单表在不同的数据库中移动。
- drop table 操作自动回收表空间。如果对于统计分析或是日值表,删除大量数据后可以通过命令"alter table TableName engine=innodb;"回收不用的空间。
- 对于使用独立表空间的表,不管怎么删除,表空间的碎片不会太严重地影响性能,而且还有机会处理。
- 单表增加过大。当单表占用空间过大时,存储空间会不足。

4. 共享表空间和独立表空间之间的转换

(1) 查看当前数据库的表空间管理类型。可以通过如下命令查看。

```
mysql > show variables like "InnoDB_file_per_table";
```

对于独立表空间,如果将全局系统变量 InnoDB_file_per_table 的值设置为 on(InnoDB_file_per_table 的默认值为 off,on 代表独立表空间管理,off 代表共享表空间管理),那么之后再创建 InnoDB 存储引擎的新表,这些表的数据信息、索引信息都将保存到独立表空间文件。

(2) 修改数据库的表空间管理方式。修改 InnoDB_file_per_table 的参数值(InnoDB_file_per_table=1 为使用独占表空间,InnoDB_file_per_table=0 为使用共享表空间)即可,

但是修改不能影响之前已经使用过的共享表空间和独立表空间。

(3) 共享表空间转化为独立表空间的方法(参数 InnoDB_file_per_table=1 需要设置)。单个表的转换操作可以用如下命令实现：

```
alter table table_name engine = innodb;
```

4.1.2 创建数据库表

创建数据库表

数据库创建之后，数据库是空的，是没有表的，可以用 show tables 命令查看。

1. 创建表的语法结构

表决定了数据库的结构，表是存放数据的地方，一个库需要什么表，各数据库表中有什么样的列，是要合理设计的。创建表的语法结构如下：

```
create [temporary]table[if not exists]table_name
[([column_definition], ...|[index_definition])]
[table_option][select_statement];
```

格式说明：

(1) temporary：使用该关键字表示创建临时表。

(2) if not exists：如果数据库中已存在某张表，再来创建一个同名的表，这时会出现错误信息。为避免错误信息，可以在建表前加上这一判断，只有该表目前不存在时才执行 create table 操作。

(3) table_name：要创建的表名。

(4) column_definition：字段的定义。包括指定字段名、数据类型、是否允许空值，指定默认值、主键约束、唯一性约束、注释字段名、是否为外键，以及字段类型的属性等。字段的定义具体格式描述如下：

```
column_name type [not null | null] [default default_value]
[auto_increment] [unique [key] | [primary] key]
[comment 'string'] [reference_definition]
```

其中：

① column_name：字段名。

② type：声明字段的数据类型。

③ null(not null)：表示字段是否可以是空值。

④ default：指定字段的默认值。

⑤ auto_increment：设置自增值属性，只有整型类型才能设置此属性。auto_increment 列值从 1 开始。每个表只能有一个 auto_increment 列，并且它必须被索引。

⑥ primary key：对字段指定主键约束。

⑦ unique key：对字段指定唯一性约束。

⑧ comment 'string'：注释字符串。

⑨ reference_definition：指定字段外键约束。

(5) index_definition：为表的相关字段指定索引。

(6) table_option：表的选项，存储引擎、字符集等。

(7) select_statement：用于定义表的查询语句。

2. 利用 SQL 语句创建数据表

本书的教务管理数据库 teaching 将根据第 1 章的需求分析和简化，创建 5 张表：student(学生表)、course(课程表)、score(成绩表)、teacher(教师表)和 teach_course(纽带表)。各表的结构如表 4-1 至表 4-5 所示。

【例 4-1】 按照表 4-1 所示的学生信息表结构创建 student 表。

表 4-1 student 表结构

列序号	字 段 名	类 型	取值说明	列 含 义
1	studentno	char(11)	主键	学生学号
2	sname	char(8)	否	学生姓名
3	sex	enum (2)	否	性别
4	birthdate	date	否	出生日期
5	entrance	int(3)	否	入学成绩
6	phone	varchar(12)	否	电话
7	Email	varchar(20)	否	电子信箱

程序代码如下：

```
mysql > create table if not exists student
        (
        studentno char(11) not null comment'学号',
        sname char(8) not null comment'姓名',
        sex enum('男', '女') default '男'comment'性别',
        birthdate date not null comment'出生日期',
        entrance int(3) null comment'入学成绩',
        phone varchar(12) not null comment'电话',
        Email varchar(20) not null comment'电子信箱',
        primary key (studentno)
        );
```

【例 4-2】 利用 create table 命令建立课程信息表 course，表结构如表 4-2 所示。

表 4-2 course 表结构

列序号	字 段 名	类 型	取值说明	列 含 义
1	courseno	char(6)	主键	课程编号
2	cname	char(20)	否	课程名称
3	type	char(8)	否	类别
4	period	int(2)	否	总学时
5	exp	int(2)	否	实验学时
6	term	int(2)	否	开课学期

程序代码如下：

```
mysql > create table if not exists course
        (
        courseno char(6) not null,
        cname char(6) not null,
        type char(8) not null,
        period int(2) not null,
        exp int(2) not null,
        term int(2) not null,
        primary key (courseno)
        );
```

【例 4-3】 利用 create table 命令建立学生分数表 score，表结构如表 4-3 所示。该表中主键由两个列构成。

表 4-3　score 表结构

列序号	字　段　名	类　　型	取值说明	列含义
1	studentno	char(11)	主键	学号
2	courseno	char(6)		课程编号
3	daily	float(3,1)	否	平时成绩
4	final	float(3,1)	否	期末成绩

利用 create table 语句在数据库 teaching 中建立学生分数表 score 的程序代码如下：

```
mysql > create table if not exists score
        (studentno char(11) not null,
        courseno char(6) not null,
        daily float(3,1) default 0,
        final float(3,1) default 0,
        primary key (studentno , courseno)
        );
```

【例 4-4】 利用 create table 命令建立教师信息表 teacher，表结构如表 4-4 所示。

表 4-4　teacher 表结构

列序号	字　段　名	类　　型	取值说明	列　含　义
1	teacherno	char(6)	主键	教师编号
2	tname	char(8)	否	教师姓名
3	major	char(10)	否	专业
4	prof	char(10)	是	职称
5	department	char(16)	否	院系部门

利用 create table 语句在数据库 teaching 中建立教师信息表 teacher 的程序代码如下：

```
mysql > create table if not exists teacher
        (teacherno char(6) not null comment '教师编号',
        tname char(8) not null comment'教师姓名',
        major char(10) not null comment '专业',
        prof char(10) not null comment '职称',
```

```
        department char(16) not null comment '部门',
        primary key (teacherno)
        );
```

【例 4-5】 为了完善 teaching 数据库的表间联系，创建表结构如表 4-5 所示的纽带表 teach_course。

表 4-5 teach_course 表结构

列序号	字段名	类型	取值说明	列含义
1	teacherno	nchar(6)	主键	教师编号
3	courseno	nchar(6)		课程编号

程序代码如下：

```
mysql > create table if not exists teach_course
        (teacherno char(6) not null,
        courseno char(6) not null,
        primary key (teacherno,courseno)
        );
```

说明：

(1) 主键设置。primary key 表示设置该字段为主键。如在 student 表中，primary key(studentno)表示将 studentno 字段定义为主键。在 score 表中，primary key(studentno,courseno)表示把 studentno、courseno 两个列一起作为复合主键。

(2) 添加注释。comment'学号'表示对 studentno 字段增加注释为"学号"。

(3) 字段类型的选择。sex enum('男','女')表示 sex 字段的字段类型是 enum，取值范围为'男'和'女'。对于取值固定的字段可以设置数据类型为 enum。例如，在 course 表的 type 字段表示的是课程的类型，一般是固定的几种类型。因此，也可以把该字段的定义写成：type enum('必修课','选修课')default '必修课'。

(4) 默认值的设置。default'男'表示默认值为"男"。

(5) 设置精度。score 表中的 daily float(3,1)表示精度为 4 ，小数为 1 位。

(6) 如果没有指定是 null 或是 not null，则列在创建时假定为 null。

3. 设置表的属性值自动增加



设置表的属性值自动增加

在 MySQL 数据表中，一个整数列可以拥有一个附加属性 auto_increment。auto_increment 也是一个特殊的约束条件。其主要用于为表中插入的新记录自动生成唯一的序列编码。默认的情况下，该字段的值是从 1 开始自增，也可以自定义开始值。一个数据表只能有一个字段使用 auto_increment 约束，且该字段必须为主键的一部分。auto_increment 约束的字段可以是任何整数类型(tinyint、smallint、int、bigint 等)。

设置属性值字段增加的基本语法规则如下：

```
属性名  数据类型  auto_increment
```

【例 4-6】 在 teaching 库中，创建选课表 sc，选课号 sc_no 是自动增量，选课时间默认为当前时间，其他字段分别是学号、课程号和教师号。

程序代码如下：

```
mysql > create table sc
        (sc_no int(6) not null auto_increment,
         studentno char(11) not null,
         courseno char(6) not null,
         teacherno char(6) not null,
         sc_time timestamp not null default now(),
         primary key (sc_no)
        );
```

4.1.3 查看表


查看表信息

数据表创建后，就可以用 show tables 命令查询已创建的表的情况。也可以查看表结构，即是指查看数据库中已存在的表的定义。查看表结构的语句包括 describe 语句和 show create table 语句。通过这两个语句，可以查看表的字段名、字段的数据类型、完整性约束条件等。

（1）查看已经创建的表。命令和运行结果如下：

```
mysql >show tables;
        +-----------------+
        | Tables_in_teaching  |
        +-----------------+
        | course              |
        | sc                  |
        | score               |
        | student             |
        | teach_course        |
        | teacher             |
        +-----------------+
        6 rows in set (0.00 sec)
```

（2）查看表基本结构语句 describe。MySQL 中，describe 语句可以查看表的基本定义。其中包括字段名、字段数据类型、是否为主键和默认值等。

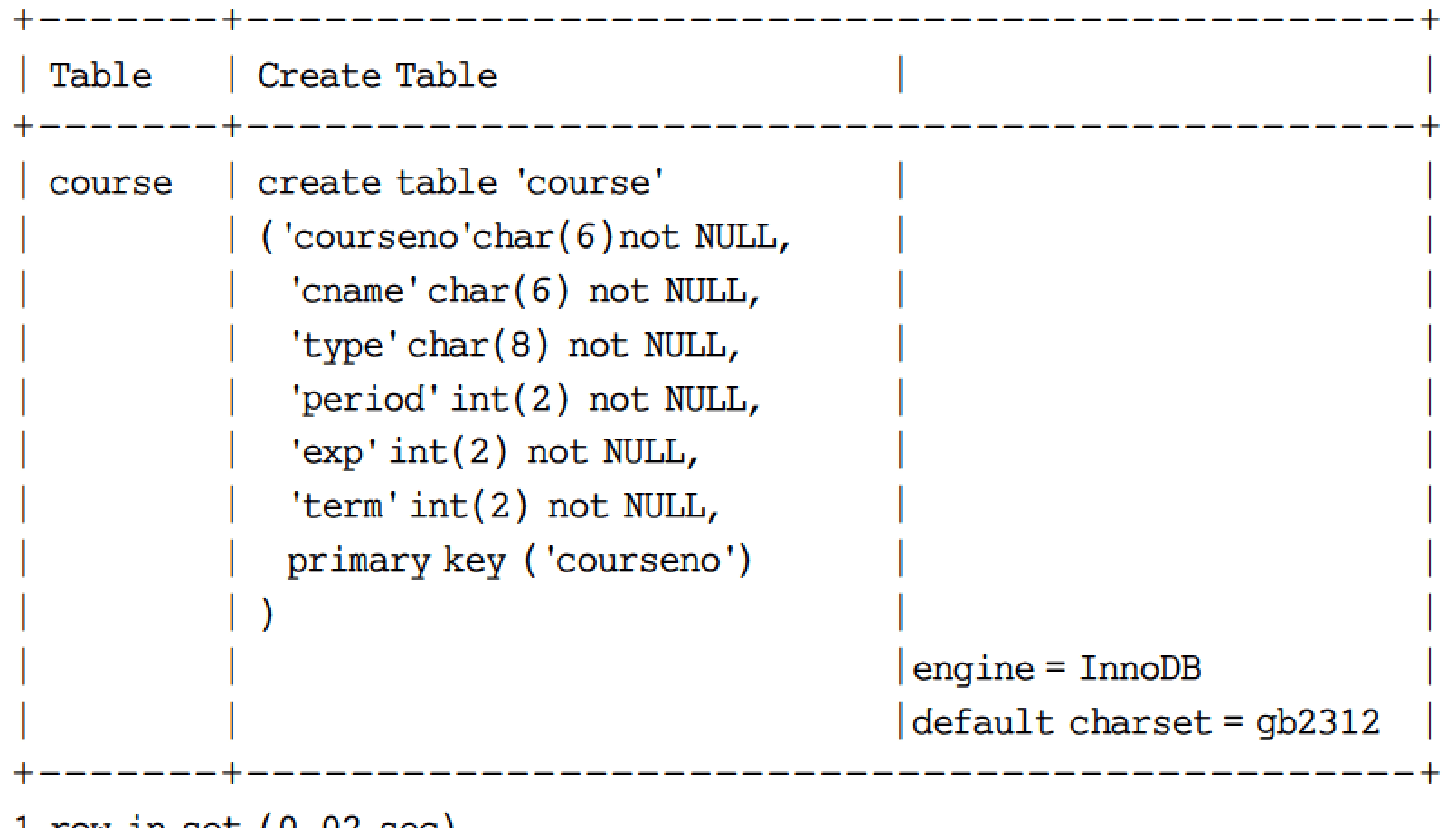
describe 语句的命令和运行结果如下：

```
mysql > describe student;
        +---------+---------------+----+----+------+-------+
        | Field      | Type             | Null | Key | Default | Extra   |
        +---------+---------------+----+----+------+-------+
        | studentno  | char(11)         | NO   | PRI | NULL    |         |
        | sname      | char(8)          | NO   |     | NULL    |         |
        | sex        | enum('男','女')  | YES  |     | 男      |         |
        | birthdate  | date             | NO   |     | NULL    |         |
        | entrance   | int(3)           | YES  |     | NULL    |         |
        | phone      | varchar(12)      | NO   |     | NULL    |         |
        | Email      | varchar(20)      | NO   |     | NULL    |         |
        +---------+---------------+----+----+------+-------+
        7 rows in set (0.05 sec)
```

(3) 查看表详细结构语句 show create table。MySQL 中，show create table 语句可以查看表的详细定义。该语句可以查看表的字段名、字段的数据类型、完整性约束条件等信息。除此之外，还可以查看表默认的存储引擎和字符编码。

show create table 语句的命令和运行结果(整理格式)如下：

```
mysql > show create table course;
```



```
+--------+--------------------------------------------------+
| Table  | Create Table                 |                    |
+--------+--------------------------------------------------+
| course | create table 'course'        |                    |
|        | ( 'courseno'char(6)not NULL, |                    |
|        |   'cname' char(6) not NULL,  |                    |
|        |   'type' char(8) not NULL,   |                    |
|        |   'period' int(2) not NULL,  |                    |
|        |   'exp' int(2) not NULL,     |                    |
|        |   'term' int(2) not NULL,    |                    |
|        |   primary key ('courseno')   |                    |
|        | )                            |                    |
|        |                              |engine = InnoDB     |
|        |                              |default charset = gb2312 |
+--------+--------------------------------------------------+
1 row in set (0.02 sec)
```

说明：

① “engine＝InnoDB”表示本表采用的存储引擎是 InnoDB，InnoDB 是 MySQL 在 Windows 平台默认的存储引擎。

② “default charset＝gb2312”表示本数据库表的字符集是 gb2312。

(4) 当数据库表创建完毕后，也可以通过安装路径(如：C:\Documents and Settings\All Users\MySQL\MySQL Server 5.7\Data\teaching)查看磁盘文件数据库及其包含的数据表文件，如图 4-1 所示。

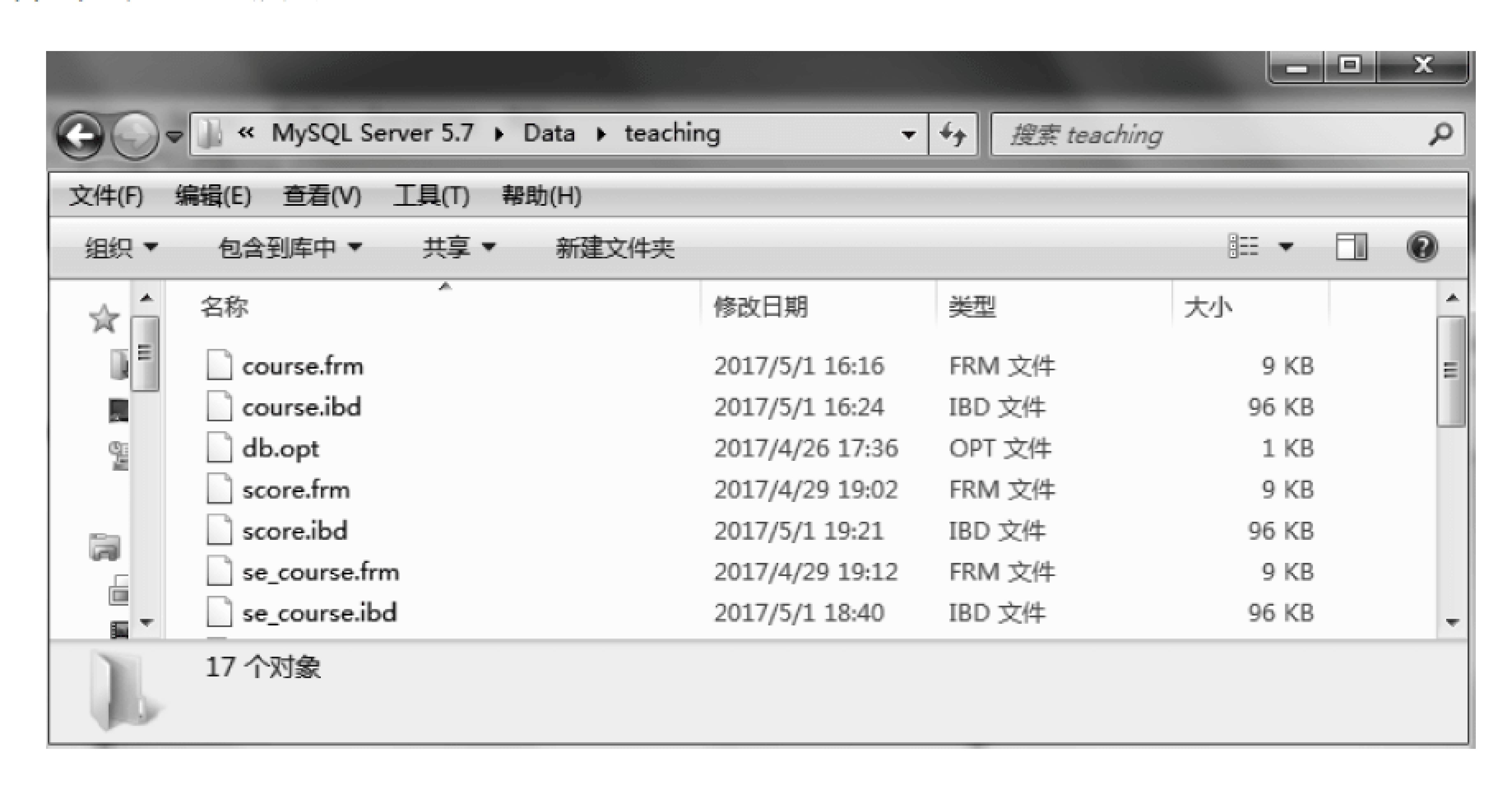


图 4-1　查看数据库表文件

从图 4-1 中可以查看数据库 teaching 中创建的各个数据表文件，如课程表 course、成绩信息表 score、选修课程表 se_course 等。

4.1.4 修改数据库表

修改表是指修改数据库中已存在的表的定义。修改表比重新定义表简单，不需要重新加载数据，也不会影响正在进行的服务。MySQL 中通过 alter table 语句来修改表。修改表包括修改表名、修改字段数据类型、修改字段名、增加字段、删除字段、修改字段的排列位置、更改默认存储引擎和删除表的外键约束等。

修改数据表

1. 修改表语法格式

修改表语法格式如下：

```
alter [ignore] table tbl_name
alter_specification [, alter_specification] ...
alter_specification:
add [column] column_definition [first | after col_name ]          //添加字段
|alter [column]col_name{set default literal | drop default}       //修改字段默认值
|change [column] old_col_name column_definition                   //重命名字段
[first | after col_name]
|modify [column]column_definition[first | aftercol_name]          //修改字段数据类型
|drop [column] col_name                                           //删除列
|rename [TO] new_tbl_name                                         //对表重命名
|order by col_name                                                //按字段排序
|convert TO character set charset_name[collate collation_name]
//将字符集转换为二进制
|[default] character set charset_name [collate collation_name]
//修改表的默认字符集
```

2. 修改表的示例

alter table 用于更改原有表的结构。例如，可以增加或删减字段、重新命名字段或表，还可以修改默认字符集。

(1) 增加字段。在创建表时，表中的字段就已经定义完成。如果要增加新的字段，可以通过 alter table 语句进行增加。增加表的字段，可以实现如下功能：

- 增加无完整性约束条件的字段。
- 增加有完整性约束条件的字段。
- 表的第一个位置增加字段。
- 表的指定位置之后增加字段。

【例 4-7】 在 student 表的 Email 列后面增加一列 address。

命令和运行结果如下：

```
mysql > alter table student
    -> add address varchar(30) not null after Email;
Query OK, 0 rows affected (1.63 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

添加操作后，也可以执行“describe student；”命令查看结果。

(2) 修改表名。表名可以在一个数据库中唯一地确定一张表。数据库系统通过表名来区分不同的表。MySQL 中，修改表名是通过 SQL 语句 alter table 实现的。

【例 4-8】 将表 sc 重命名为 se_course。

命令和运行结果如下：

```
mysql > alter table sc rename to se_course;
        Query OK, 0 rows affected (0.20 sec)
```

(3) 修改字段的数据类型。alter table 语句也可以修改字段的数据类型。

【例 4-9】 修改 course 表的 type 字段，因为该字段一般是取固定值。因此，也可以把该字段的定义写成：type enum('必修','选修')default '必修'。

命令和运行结果如下：

```
mysql > alter table course
     -> modify type enum('必修','选修') default '必修';
      Query OK, 0 rows affected (0.47 sec)
      Records: 0 Duplicates: 0 Warnings: 0
```

(4) 删除字段。删除字段是指删除已经定义好的表中的某个字段。MySQL 中，alter table 语句也可以删除表中的字段。

【例 4-10】 删除 student 表的字段 address。

命令和运行结果如下：

```
mysql > alter table student drop address;
        Query OK, 0 rows affected (0.21 sec)
        Records: 0 Duplicates: 0 Warnings: 0
```

4.1.5 删除数据库表

删除表是指删除数据库中已存在的表。删除表时，会删除表中的所有数据。因此，在删除表时要特别注意。MySQL 中通过 drop table 语句来删除表。

删除表的语法格式如下：

```
drop table table_name
```



管理数据表

【例 4-11】 在 mysqltest 数据库中创建表 example，然后删除 example 表。

代码和运行结果如下：

```
mysql > use mysqltest;
        Database changed
mysql > Create table example(
     -> today datetime,
     -> name char(20)
     -> );
         Query OK, 0 rows affected (0.11 sec)
mysql > desc example;
        +--------+---------+------+-----+--------+--------+
        | Field  | Type    | Null | Key | Default | Extra |
        +--------+---------+------+-----+--------+--------+
        | today  | datetime | YES |     | NULL   |        |
        | name   | char(20) | YES |     | NULL   |        |
        +--------+---------+------+-----+--------+--------+
        2 rows in set (0.02 sec)
```

```
mysql > drop table example ;
        Query OK, 0 rows affected (0.07 sec)
```

代码运行成功，从数据库 mysqltest 中删除 example 表。在执行代码之前，先用 desc 语句查看是否存在 example 表，以便与删除后进行对比。

4.1.6 临时表的管理

MySQL 临时表适合当工作在非常大的表上时，偶尔需要运行很多查询获得一个大量数据的小的子集，不是对整个表运行这些查询，而是让 MySQL 每次找出所需的少数记录，将记录选择到一个临时表可能更快些，然后对这些表运行查询。

创建临时表很容易，给正常的 create table 语句加上 temporary 关键字即可。例如，创建临时表 tmp_emp1：

```
mysql > create temporary table tmp_emp1
    -> (name varchar(10) not null,
    -> value integer not null
    -> );
```

临时表将在连接 MySQL 期间存在。断开时，MySQL 将自动删除表并释放所用的空间。当然也可以在仍然连接的时候删除临时表并释放空间。删除方法与一般用户表相同：

```
drop table tmp_table
```

说明：

(1) 创建临时表必须有 create temporary table 权限。

(2) show tables 语句不会列举临时表。

(3) 不能用 rename 来重命名一个临时表。

4.2 表的数据操作

MySQL 数据表分为表结构(Structure)和数据记录(Record)两部分。前面创建表的操作，仅仅是创建了表结构，表结构即决定表拥有哪些字段以及这些字段的名称、数据类型、长度、精度、小数位数、是否允许空值(null)、设置默认值和主键等。而表数据的操作将在本节介绍。

MySQL 语言一般通过 insert、update 和 delete 三种 DML 语句对表进行数据的添加、更新和删除数据操作，并以此维护和修改表的数据。

4.2.1 表记录的插入

为数据表输入数据的方式有多种，常见的有通过命令方式添加行数据的，也可以通过程序实现表数据的添加。可以通过 insert、replace 语句插入，也可以使用 load data infile 方式将保存在文本文件中的数据插入到指定的表。



insert 语句

1. 使用 insert|replace 语句添加数据

insert|replace 语句语法格式：

```
insert|replace[into]table_name[(col_name,...)]
values({expr|default},...),(...),...
```

```
|set col_name = {expr|default}, ...
```

【例 4-12】 利用 insert 命令向表 student 中插入一行数据。

代码和运行结果如下：

```
mysql> insert into student
    ->(studentno,sname,sex,birthdate,entrance,phone,Email)
    -> values ('18122210009','许东山','男','1999/11/5',789,
    -> '13623456778','qwe@163.com');
     Query OK, 1 row affected (0.07 sec)
```

【例 4-13】 利用 insert 命令向表 student 中插入多行数据。

代码和运行结果如下：

```
mysql> insert into student values
    -> ('18122221324','何白露',
    -> '女','2000/12/4','879','13178978999','heyy@sina.com '),
    -> ('18125111109','敬横江',
    -> '男','2000/3/1','789','15678945623','jing@sina.com '),
    -> ('18125121107','梁一苇',
    -> '女','1999/9/3','777','13145678921','bing@126.com '),
    -> ('18135222201','凌浩风',
    -> '女','2001/10/6','867','15978945645','tang@163.com '),
    -> ('18137221508','赵临江',
    -> '男','2000/2/13','789','12367823453','ping@163.com '),
    -> ('19111133071','崔依歌',
    -> '女','2001/6/6','787','15556845645','cui@126.com '),
    -> ('19112100072','宿沧海',
    -> '男','2002/2/4','658','12545678998','su12@163.com'),
    -> ('19112111208','韩山川',
    -> '男','2001/2/14','666','15878945612','han@163.com '),
    -> ('19122203567','封月明',
    -> '女','2002/9/9','898','13245674564','jiao@126.com'),
    -> ('19123567897','赵既白',
    -> '女','2002/8/4','999','13175689345','pingan@163.com'),
    -> ('19126113307','梅惟江',
    -> '女','2003/9/7','787','13245678543','zhu@163.com');
   Query OK, 11 rows affected (0.05 sec)
```

【例 4-14】 利用 replace 命令向表 course 中插入多行数据。

代码和运行结果如下：


replace 语句

```
mysql> replace into course values
    -> ('c05103','电子技术','必修','64','16','2'),
    -> ('c05109','C 语言','必修','48','16','2'),
    -> ('c05127','数据结构','必修','64','16','2'),
    -> ('c05138','软件工程','选修','48','8','5'),
    -> ('c06108','机械制图','必修','60','8','2'),
    -> ('c06127','机械设计','必修','64','8','3'),
    -> ('c06172','铸造工艺','选修','42', '16','6'),
    -> ('c08106','经济法','必修','48','0','7'),
    -> ('c08123','金融学','必修','40','0','5'),
```

```
    ->('c08171','会计软件','选修','32','8','8');
   Query OK, 10 rows affected (0.05 sec)
   Records: 10 Duplicates: 0 Warnings: 0
```

说明：

(1) 使用 insert 语句可以向表中插入一行数据，也可以插入多行数据，最好一次插入多行数据，各行数据之间用“，”分隔。

(2) values 子句：包含各列需要插入的数据清单，数据的顺序要与列的顺序相对应。若表名后不给出列名，则在 values 子句中要给出每一列（除 identity 和 timestamp 类型的列）的值，如果列值为空，则值必须置为 null，否则会出错。

(3) 如果向表中添加已经存在的学号（已经设为主键）的记录，因此将出现主键冲突错误。例如，插入已经存在的学号 19112111208 记录，结果如下：

```
mysql> insert into student values('19112111208','韩小雨',
    ->'女','2001/2/14','666','15878945612','han@163.com ');
     ERROR 1062 (23000): Duplicate entry '19112111208' for key 'primary'
```

(4) 用 replace 向表中插入数据时，首先尝试插入数据到表中，如果发现表中已经有此行数据（根据主键或者唯一索引判断），则先删除此行数据，然后插入新的数据，否则，直接插入新数据。

(5) 还可以向表中插入其他表的数据，这也是成批插入数据的一种方式。但要求两个表要有相同的结构。具体操作将在以后介绍。其语法格式如下：

```
insert into table name1 select * from table name2;
```

2. 利用 load data 语句将数据装入数据库表中

【例 4-15】 假设 teacher 表的数据已放在“d:\teacher.txt”中，现将 teaching.txt 的数据插入到 teacher 表中。

代码和运行结果如下：

load data 语句

```
mysql>load data local infile "d:\\teacher.txt" into table teacher;
       Query OK, 9 rows affected, 8 warnings (0.03 sec)
       Records: 9 Deleted: 0 Skipped: 0 Warnings: 0
mysql> select * from teacher;          //输出表的记录
        +---------+--------+----------+--------+------------+
        | teacherno | tname | major    | prof   | department |
        +---------+--------+----------+--------+------------+
        | t05001  | 苏超然 | 软件工程 | 教授   | 计算机学院 |
        | t05002  | 常杉   | 会计学   | 助教   | 管理学院   |
        | t05003  | 孙释安 | 网络安全 | 教授   | 计算机学院 |
        | t05011  | 卢敖治 | 软件工程 | 副教授 | 计算机学院 |
        | t05017  | 茅佳峰 | 软件测试 | 讲师   | 计算机学院 |
        | t06011  | 夏南望 | 机械制造 | 教授   | 机械学院   |
        | t06023  | 葛庭宇 | 铸造工艺 | 副教授 | 材料学院   |
        | t07019  | 韩既乐 | 经济管理 | 讲师   | 管理学院   |
        | t08017  | 时观   | 金融管理 | 副教授 | 管理学院   |
        +---------+--------+----------+--------+------------+
        9 rows in set (0.00 sec)
```

说明：

(1) teacher.txt 各行文本之间要用制表符< Tab >分隔，每行最后也加< Tab >分隔符。

(2) "d:\\teacher.txt"：要用"\\"，表示斜线。

(3) 以此类推，可以将 score 表和 teach_course 表的数据载入。

【例 4-16】 利用 load data 语句输入 score 表数据。

代码和运行结果如下：

```
mysql >load data local infile "d:\\score.txt" into table score;
         Query OK, 28 rows affected, 27 warnings (0.07 sec)
         Records: 28 Deleted: 0 Skipped: 0 Warnings: 0
mysql > select * from score;
          +-----------+---------+-------+-------+
          | studentno | courseno | daily | final |
          +-----------+---------+-------+-------+
          | 18122210009 | c05103 | 87.0 | 82.0 |
          | 18122210009 | c05109 | 77.0 | 91.0 |
          | 18122221324 | c05103 | 88.0 | 62.0 |
          | 18122221324 | c05109 | 91.0 | 77.0 |
          | 18125111109 | c08106 | 79.0 | 99.0 |
          | 18125111109 | c08123 | 85.0 | 92.0 |
          | 18125111109 | c08171 | 77.0 | 92.0 |
          | 18125121107 | c05103 | 74.0 | 91.0 |
          | 18125121107 | c05109 | 89.0 | 62.0 |
          | 18135222201 | c05109 | 99.0 | 92.0 |
          | 18135222201 | c08171 | 95.0 | 82.0 |
          | 18137221508 | c08106 | 78.0 | 95.0 |
          | 18137221508 | c08123 | 78.0 | 89.0 |
          | 18137221508 | c08171 | 88.0 | 98.0 |
          | 19111133071 | c05103 | 82.0 | 69.0 |
          | 19111133071 | c05109 | 77.0 | 82.0 |
          | 19112100072 | c05109 | 87.0 | 86.0 |
          | 19112100072 | c06108 | 97.0 | 97.0 |
          | 19112111208 | c05109 | 85.0 | 91.0 |
          | 19112111208 | c06108 | 89.0 | 95.0 |
          | 19122111208 | c06127 | 78.0 | 67.0 |
          | 19122203567 | c05103 | 65.0 | 98.0 |
          | 19122203567 | c05108 | 88.0 | 89.0 |
          | 19122203567 | c06127 | 79.0 | 88.0 |
          | 19123567897 | c05103 | 85.0 | 77.0 |
          | 19123567897 | c06127 | 99.0 | 99.0 |
          | 19126113307 | c06108 | 66.0 | 82.0 |
          | 19126113307 | c08171 | 88.0 | 79.0 |
          +-----------+---------+-------+-------+
          28 rows in set (0.00 sec)
```

【例 4-17】 利用 load data 语句输入 teach_course 表数据。

代码和运行结果如下：

```
mysql > load data local infile "d:\\teach_course.txt"
```

```
    -> into table teach_course;
      Query OK, 9 rows affected, 8 warnings (0.03 sec)
      Records: 9 Deleted: 0 Skipped: 0 Warnings: 0
mysql> select * from teach_course;
    +----------+--------+
    | teacherno | courseno |
    +----------+--------+
    | t05001    | c05109   |
    | t05002    | c05127   |
    | t05003    | c05127   |
    | t05011    | c05138   |
    | t05017    | c05127   |
    | t06011    | c06127   |
    | t06023    | c06172   |
    | t07019    | c08123   |
    | t08017    | c08106   |
    +----------+--------+
    9 rows in set (0.00 sec)
```

3. 使用 set 子句插入数据

用 set 子句直接赋值时可以不按列顺序插入数据，对允许空值的列可以不插入。

【例 4-18】 利用 set 子句向 se_course 表插入数据。

代码和运行结果如下：

```
mysql> insert into se_course
    -> set studentno='19120000111',courseno='c01236',teacherno='t01237';
      Query OK, 1 row affected (0.04 sec)
mysql> select * from se_course;
     +-------+-----------+--------+---------+-------------------+
     | sc_no | studentno   | courseno | teacherno | sc_time             |
     +-------+-----------+--------+---------+-------------------+
     |     1 | 19120000111 | co1236   | t01237    | 2017-05-01 18:40:23 |
     +-------+-----------+--------+---------+-------------------+
     1 row in set (0.00 sec)
```

4. 图片数据的插入


图片数据的插入

MySQL 还支持图片的存储，图片一般可以以路径的形式来存储，即插入图片采用直接插入图片的存储路径。当然，也可以直接插入图片本身，只要用 load_file( )函数即可。

【例 4-19】 参照 student 表结构创建 student01 表，添加一个能够存储图片的字段，然后插入一行数据。照片路径为“d:\image\picture.jpg”。

代码和运行结果如下：

```
mysql> create table student01 as select * from student;
    Query OK, 12 rows affected (0.29 sec)
    Records: 12 Duplicates: 0 Warnings: 0
mysql> select * from student01;
    (结果与 student 表一样,从略)
    12 rows in set (0.00 sec)
mysql> alter table student01 add fields mediumblob comment'照片';
```

```
    Query OK, 0 rows affected (0.29 sec)
    Records: 0 Duplicates: 0 Warnings: 0
mysql> insert into student01 values
    -> ('18122221329','何影映','女','2001/12/9','877',
    -> '13178978997', 'heyy1@sina.com ', 'd:\\image\\picture.jpg');
    Query OK, 1 row affected (0.04 sec)
```

下面语句是直接存储图片本身：

```
mysql> insert into student01 values
    -> ('18122221329','何影映','女','2001/12/9','877', '13178978997',
    -> 'heyy1@sina.com ', load_file('d:\\image\\picture.jpg');
```

说明：

(1) 存放图片的字段要使用 blob 类型。blob 是专门存储二进制文件的类型，有大小之分，例如 mediumblob、longblob 等，以存储大小不同的二进制文件，一般的图形文件使用 mediumblob 就足够了。

(2) 插入图片文件路径的办法要比插入图片本身好。图片如果很小的话，可以存入数据库，但是如果图片大的话，保存或读取操作会很慢，倒不如将图片存入指定的文件夹，然后把文件路径和文件名存入数据库。

4.2.2 表记录的修改


表记录的修改

用 update...set...命令可以修改一个表的数据。一般表记录修改的语法格式如下：

```
update table_name
set col_name1 = [,col_name2 = expr2 ...]
[where 子句]
```

说明：

(1) set 子句：根据 where 子句中指定的条件，对符合条件的数据行进行修改。若语句中不设定 where 子句，则更新所有行。

(2) expr1、expr2、...：可以是常量、变量或表达式。可以同时修改所在数据行的多个列值，中间用逗号隔开。

【例 4-20】 将学号为 18137221508 的学生的课程号为 c08106 的平时成绩 daily 修改为 80 分。命令和运行结果如下：

```
mysql> update score set daily = 80
    -> where studentno = '18137221508' && courseno = 'c08106';
       Query OK, 1 row affected (0.03 sec)
       Rows matched: 1 Changed: 1 Warnings: 0
mysql> select * from score
    -> where studentno = '18137221508' && courseno = 'c08106';
       +-------------+----------+--------+--------+
       | studentno   | courseno | daily  | final  |
       +-------------+----------+--------+--------+
       | 18137221508 | c08106   | 80.0   | 95.0   |
       +-------------+----------+--------+--------+
       1 row in set (0.01 sec)
```

【例 4-21】 将课程 student01 表中低于 700 分的入学成绩增加 8%。

命令和运行结果如下：

```
mysql > update student01 set entrance = entrance * 1.08 where entrance < 700;
        Query OK, 2 rows affected (0.04 sec)
        Rows matched: 2 Changed: 2 Warnings: 0
```

4.2.3 表记录的删除



表记录的删除

利用 delete...from...语句可以从单个表中删除指定表数据，一般表记录删除的语法格式如下：

```
delete[low_priority] [quick] [ignore] from tbl_name
[where 子句]
[order by 子句]
[limit row_count]
```

说明：

(1) low_priority：用来降低删除优先级，让查询操作优先。

(2) quick 修饰符：可以加快部分种类的删除操作的速度。

(3) from 子句：用于指定从何处删除数据。

(4) ignore 修饰符：忽略删除错误，提高运行速度。

(5) where 子句：指定的删除条件。如果省略 where 子句则删除该表的所有行。

(6) order by 子句：各行按照子句中指定的顺序进行删除，此子句只在与 limit 联用时才起作用。

(7) limit 子句：用于告知服务器在控制命令被返回到客户端前被删除的行的最大值。

(8) 数据删除后将不能恢复，因此，在执行删除之前一定要对数据做好备份。

【例 4-22】 删除 student01 表中入学成绩低于 750 分的记录。

命令和运行结果如下：

```
mysql > delete from student01 where entrance < 750;
        Query OK, 2 rows affected (0.04 sec)
```

【例 4-23】 删除 student01 表中入学成绩最低的 2 行记录。

命令和运行结果如下：

```
mysql > delete from student01 order by entrance limit 2;
        Query OK, 2 rows affected (0.01 sec)
```

4.3 利用 MySQL Workbench 管理表

MySQL Workbench 为数据库管理员、程序开发者和系统规划师提供可视化设计、模型建立以及数据库管理功能。Workbench 包含了用于创建复杂的数据建模 E-R 模型，正向和逆向数据库工程，也可以用于执行通常需要花费大量时间和难以变更和管理的文档任务。MySQL 工作台可在 Windows、Linux 和 Mac 上使用。

利用 Workbench 管理数据库和表为初学者学习 MySQL 提供了非常可靠、简单、方便的可视化操作工具。

4.3.1 数据表的创建



利用 Workbench 创建表

创建 MySQL 数据表，前面采用的是利用命令的方式，在此采用可视化方式。下面介绍利用可视化方式创建表的过程。

(1) 打开 MySQL Workbench 工具。在如图 4-2 所示的导航区 Navigator 下的 Schemas 区域，可以看到前面创建的数据库 teaching 中的各个数据表。此时，可以看到 teaching 字体是粗体，颜色较深，是当前数据库。

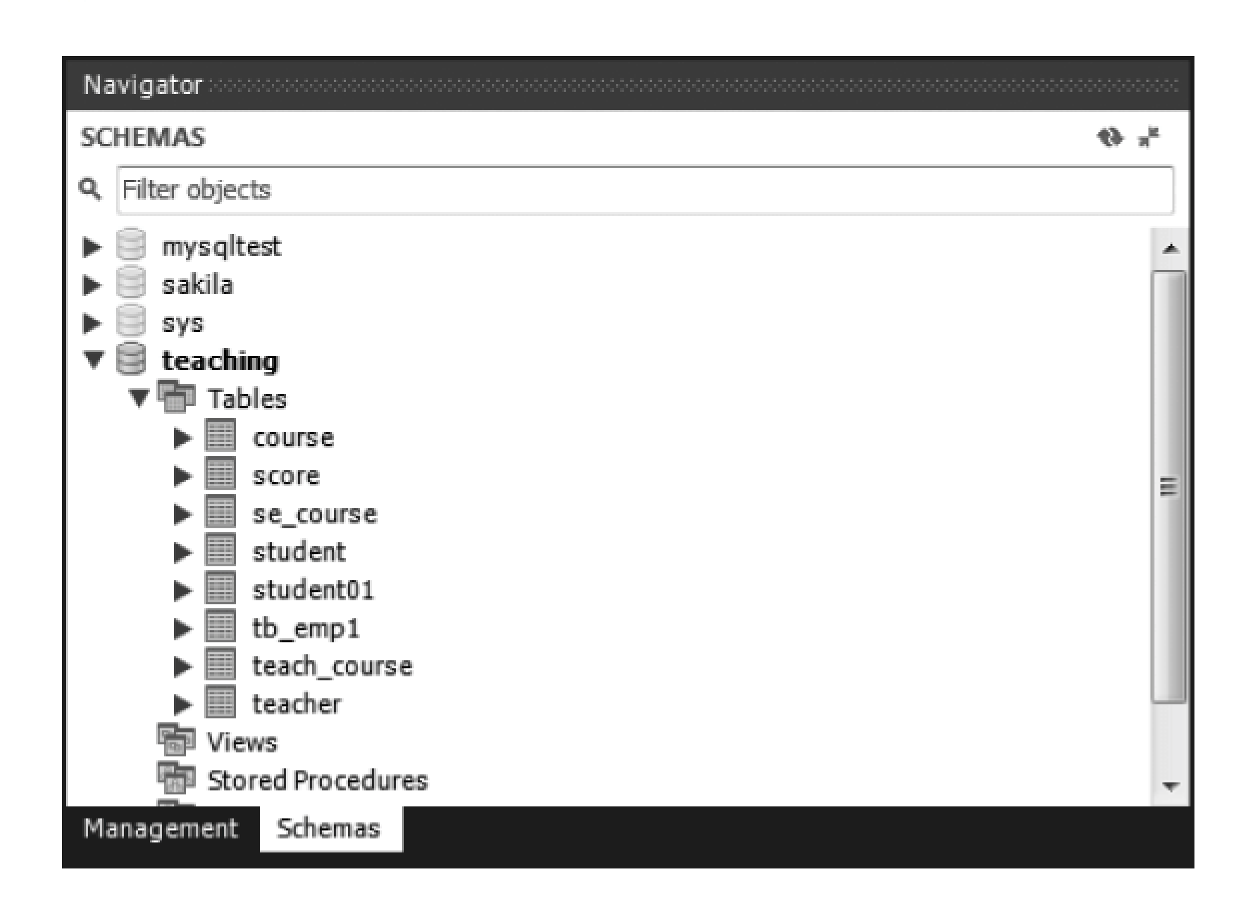


图 4-2 可视化方式查看数据表

(2) 选择 mysqltest 数据库为当前数据库。右击 mysqltest 数据库，如图 4-3 所示。在弹出的菜单中执行 Set as Default Schema 命令，mysqltest 数据库就变成当前数据库。

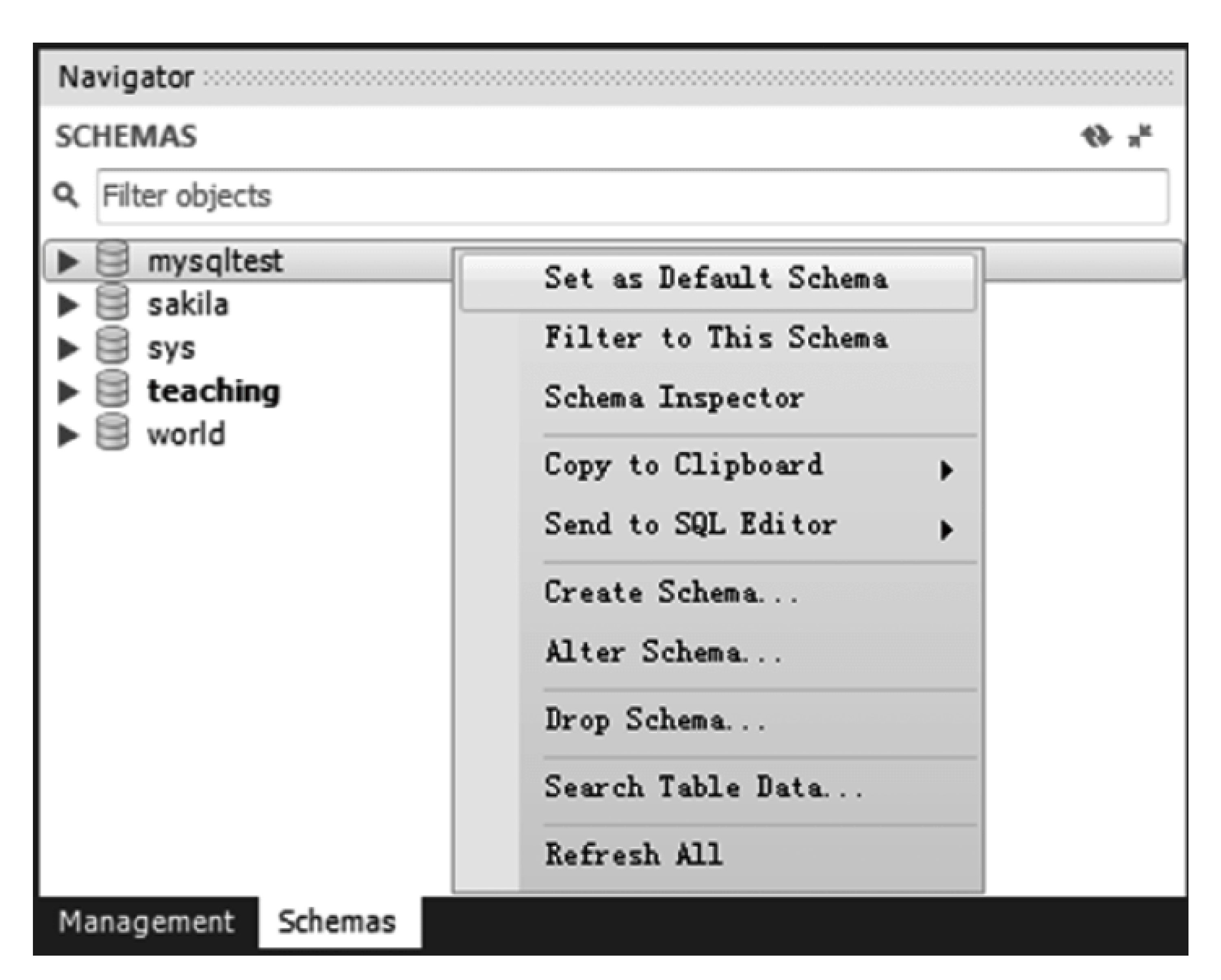


图 4-3 选择当前数据库

(3) 在 mysqltest 数据库中选择 Tables，右击 Tables 选项，在如图 4-4 所示的弹出菜单中执行 Create Table 命令。进入如图 4-5 所示的数据表创建初始界面。

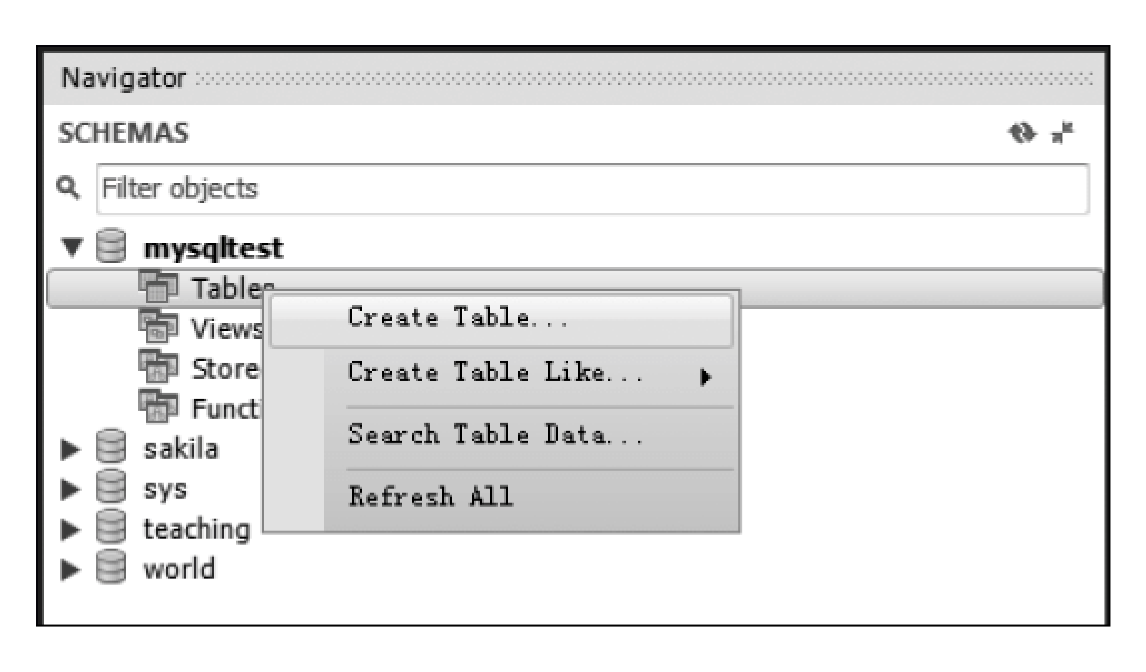


图 4-4　创建数据表操作命令

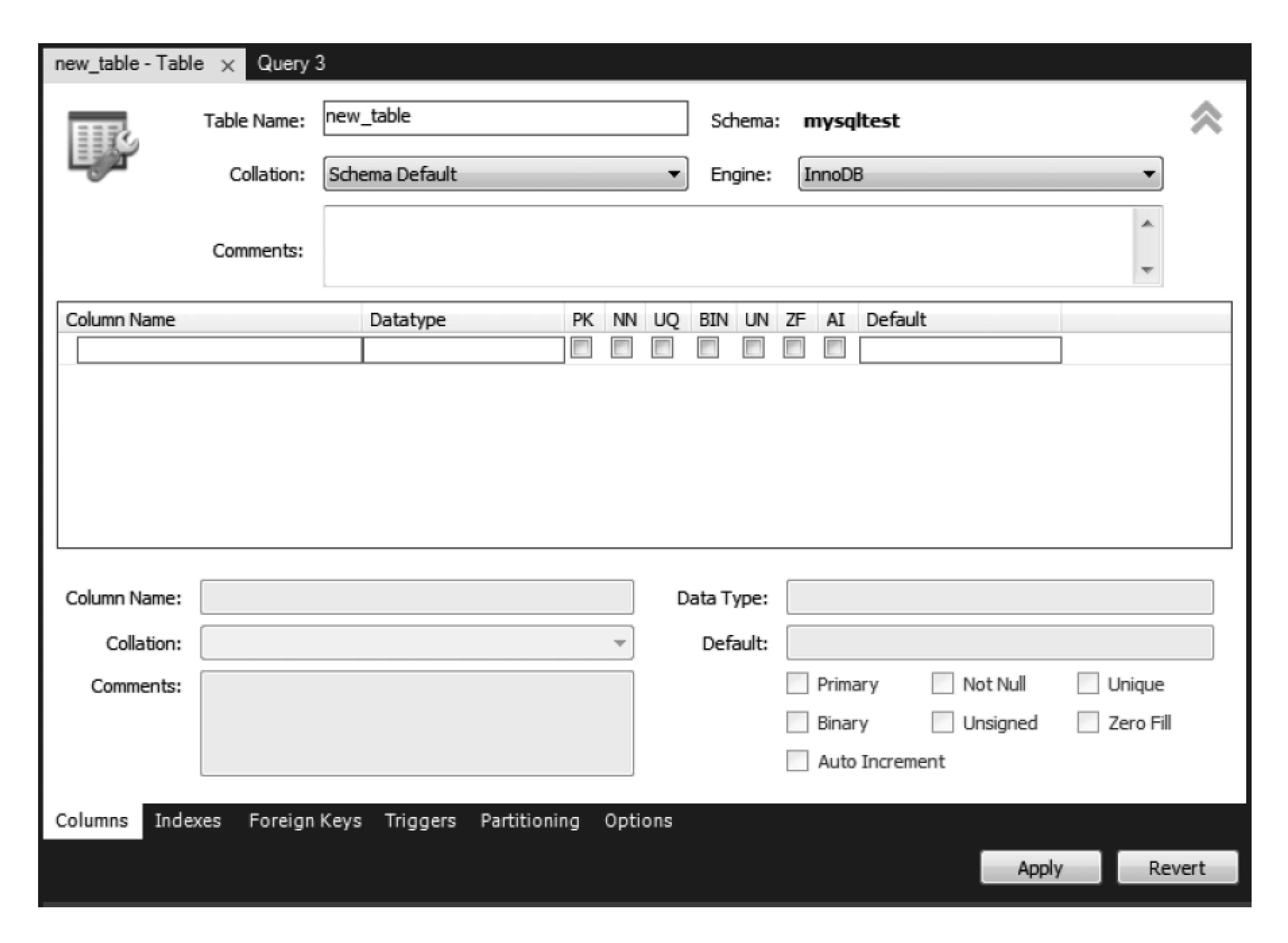


图 4-5　创建数据表初始界面

(4) 输入数据库表名 student02，选择默认数据库引擎 InnoDB，参照前面的表 4-1 所示的 student 表结构，创建数据表 student02。分别输入列名，选择数据类型，输入不同类型的参数或默认值，如图 4-6 所示。

(5) 单击 Apply 按钮，进入如图 4-7 所示的脚本审核对话框。可以再次编辑创建数据表的文本。单击 Apply 按钮，进入完成界面。单击 Show Logs 按钮，可以进一步查看脚本，如图 4-8 所示。

(6) 单击 Finish 按钮，完成数据表 student02 的创建。展开数据库 mysqltest 数据库中的 Tables 文件夹，即可查看到表 student02 及其表结构。

图 4-6　输入数据表结构参数

```
CREATE TABLE `mysqltest`.`student02` (
  `studentno` CHAR(11) NOT NULL,
  `sname` CHAR(8) NULL,
  `sex` ENUM('男','女') NULL DEFAULT '女',
  `birthdate` DATE NULL,
  `entrance` INT(3) NULL,
  `phone` VARCHAR(12) NULL,
  `Email` VARCHAR(20) NULL,
  PRIMARY KEY (`studentno`));
```

图 4-7　编辑创建表的脚本语句



图 4-8　完成数据表的创建

4.3.2　编辑数据



利用 Workbench 编辑数据

数据表创建成功后，就可以实现对 mysqltest 数据库的 student02 数据表结构进行修改、删除等操作，如图 4-9 所示。也可以编辑、查询和添加数据。下面进行数据表的输入数据操作。

(1) 在图 4-9 所示的页面中，选择 Select Rows-Limit 1000 命令，就可以进入如图 4-10 所示的查询窗体。在该对话框中，不但可以进行数据添加，还可以执行查询、拷贝数据操作，也可以修改表结构和删除表格等操作。自己可以对相关工具栏按钮进行数据的添加删除操作。

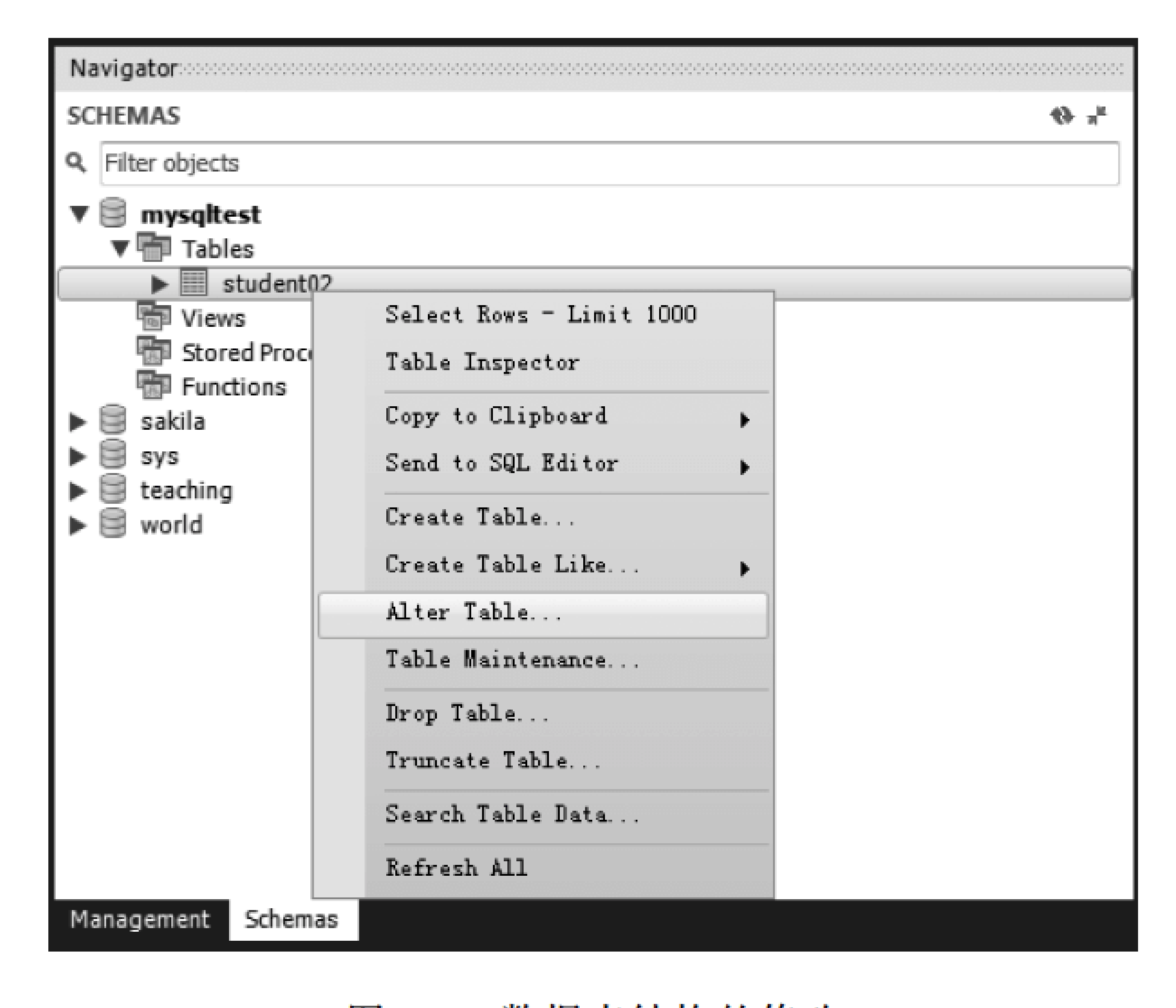


图 4-9　数据表结构的修改



图 4-10　编辑表数据

（2）按照如图 4-11 中所示的示例，可以向数据 student02 表中添加数据。添加完毕，单击 Apply 按钮，进入如图 4-12 所示的脚本审核界面。

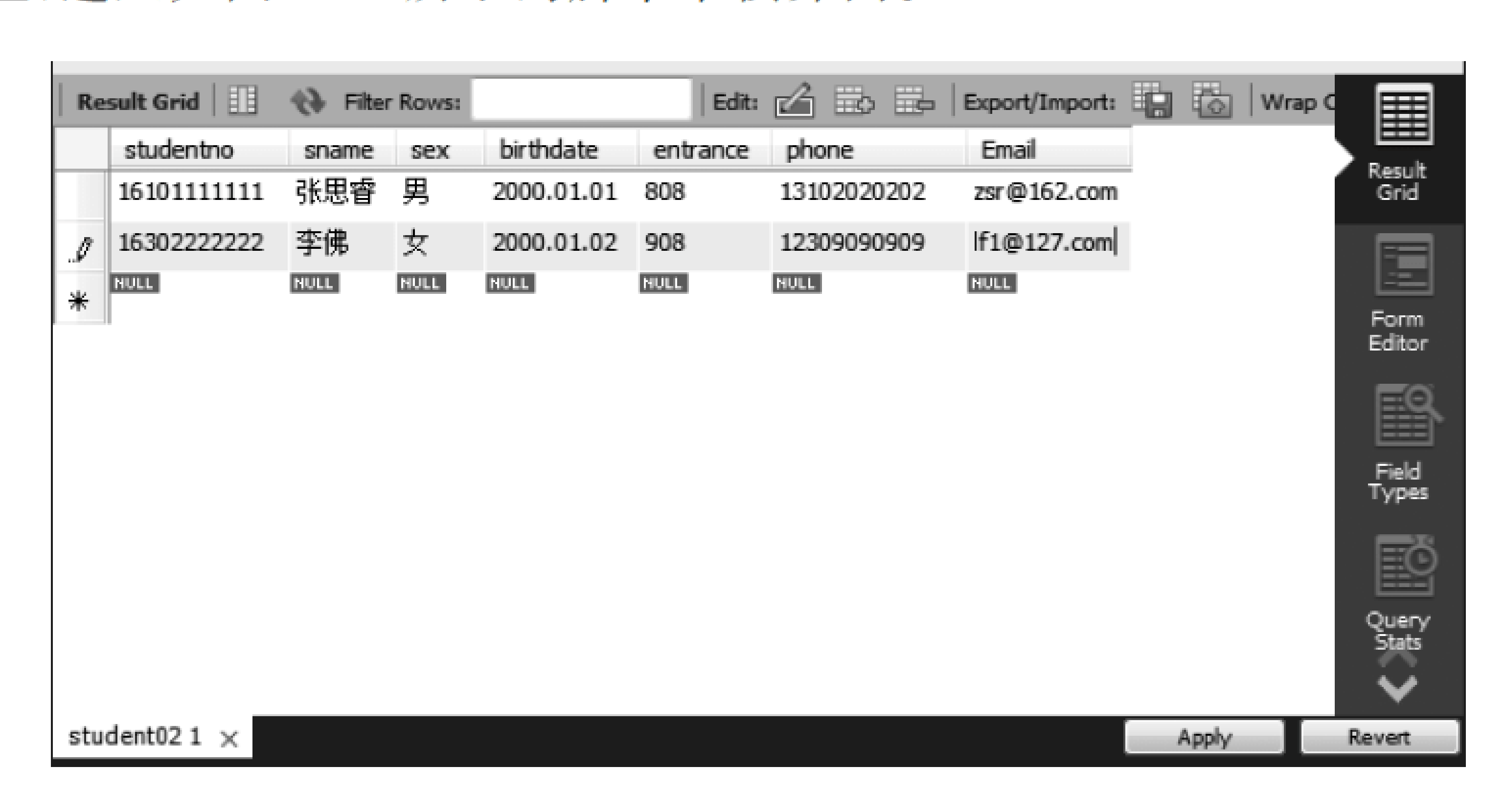


图 4-11　输入表数据

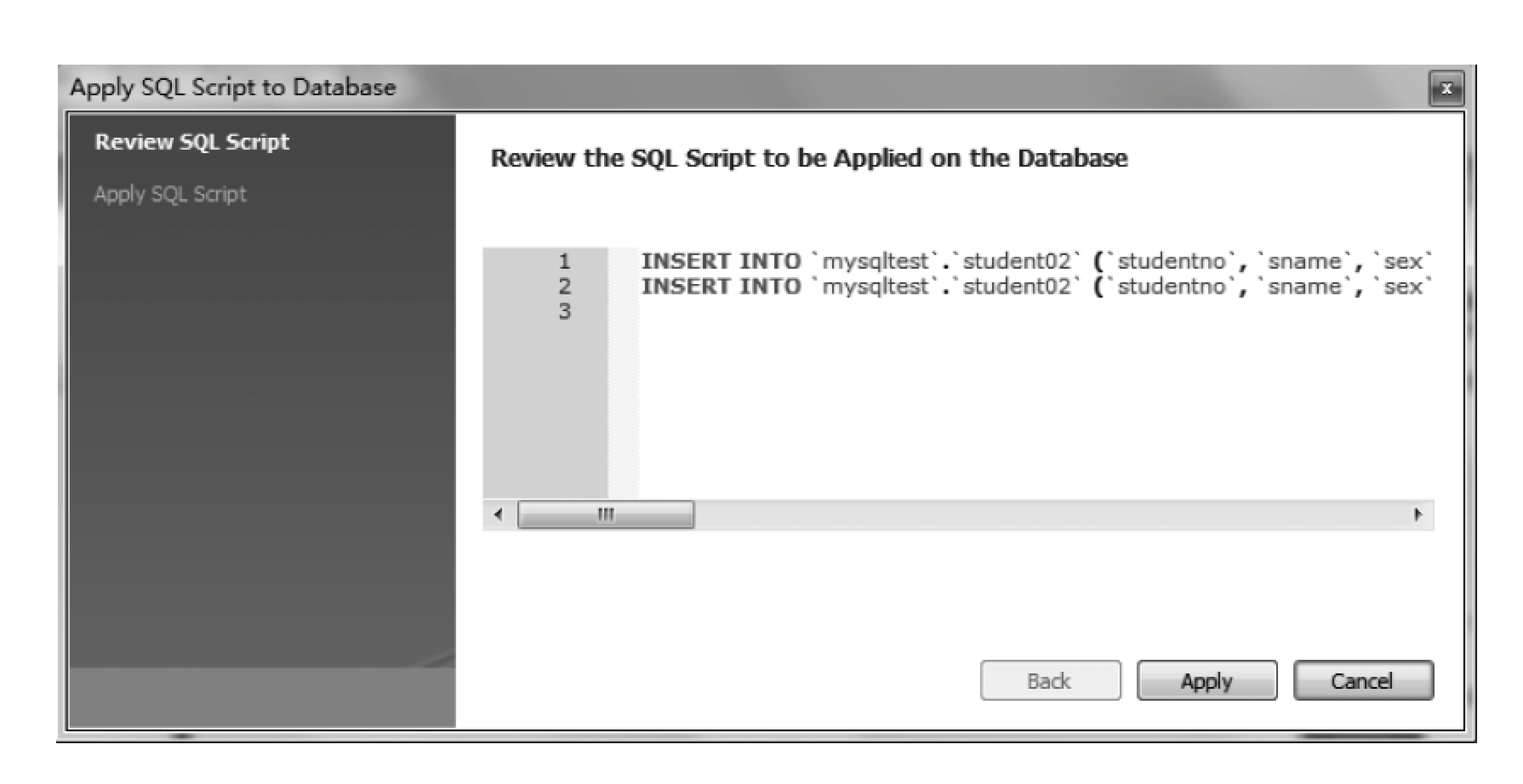


图 4-12　输入表数据操作的脚本

(3) 单击 Apply 按钮,打开如图 4-13 所示的界面,单击 Show Logs 按钮,可以进一步查看插入数据的脚本。

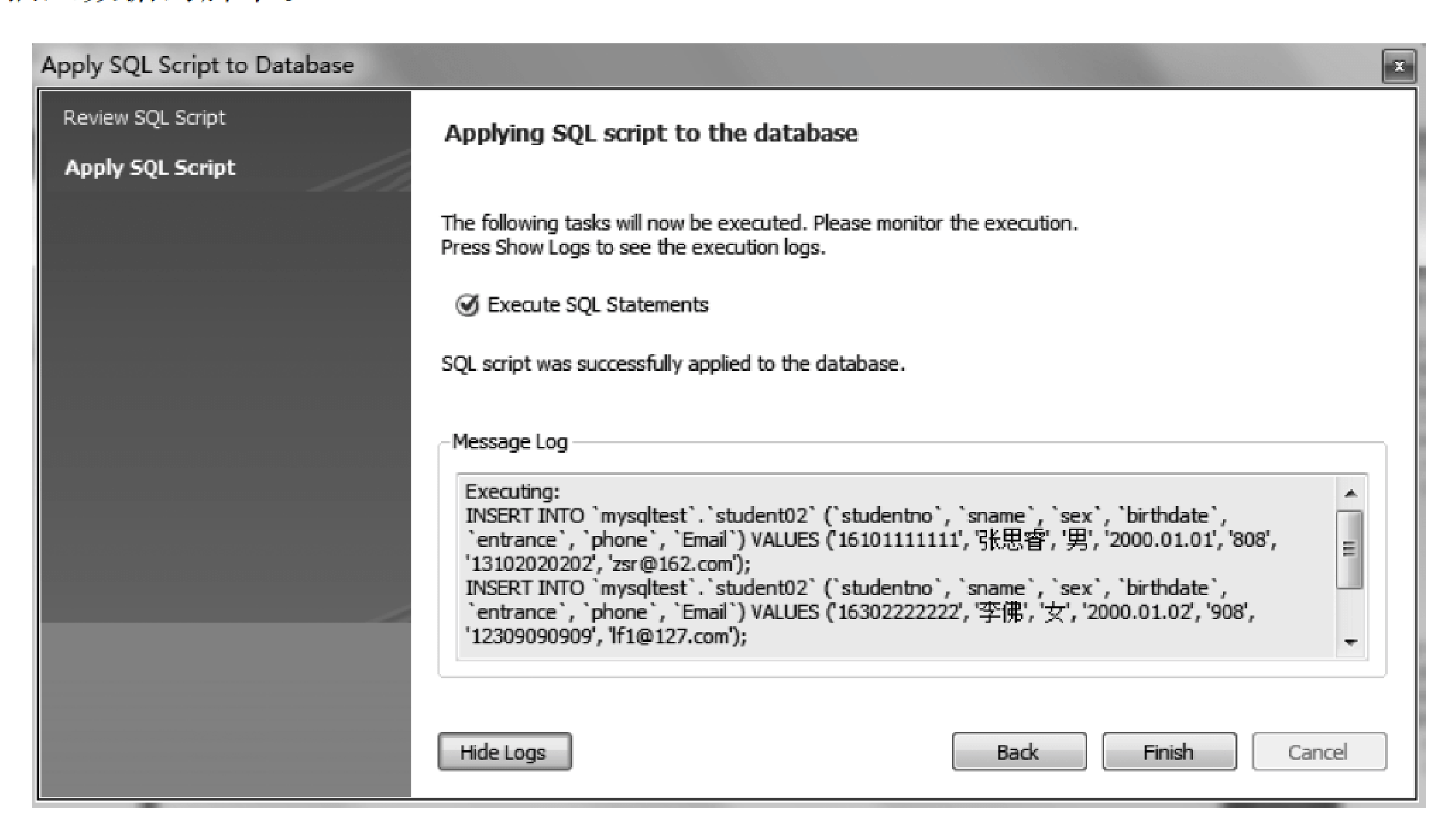


图 4-13　完成输入表数据操作

(4) 单击 Finish 按钮,完成数据表 student02 的数据输入过程。

(5) 数据添加完成后,单击如图 4-11 中的 Form Editor 图标,还可以继续执行数据的添加操作,如图 4-14 所示。

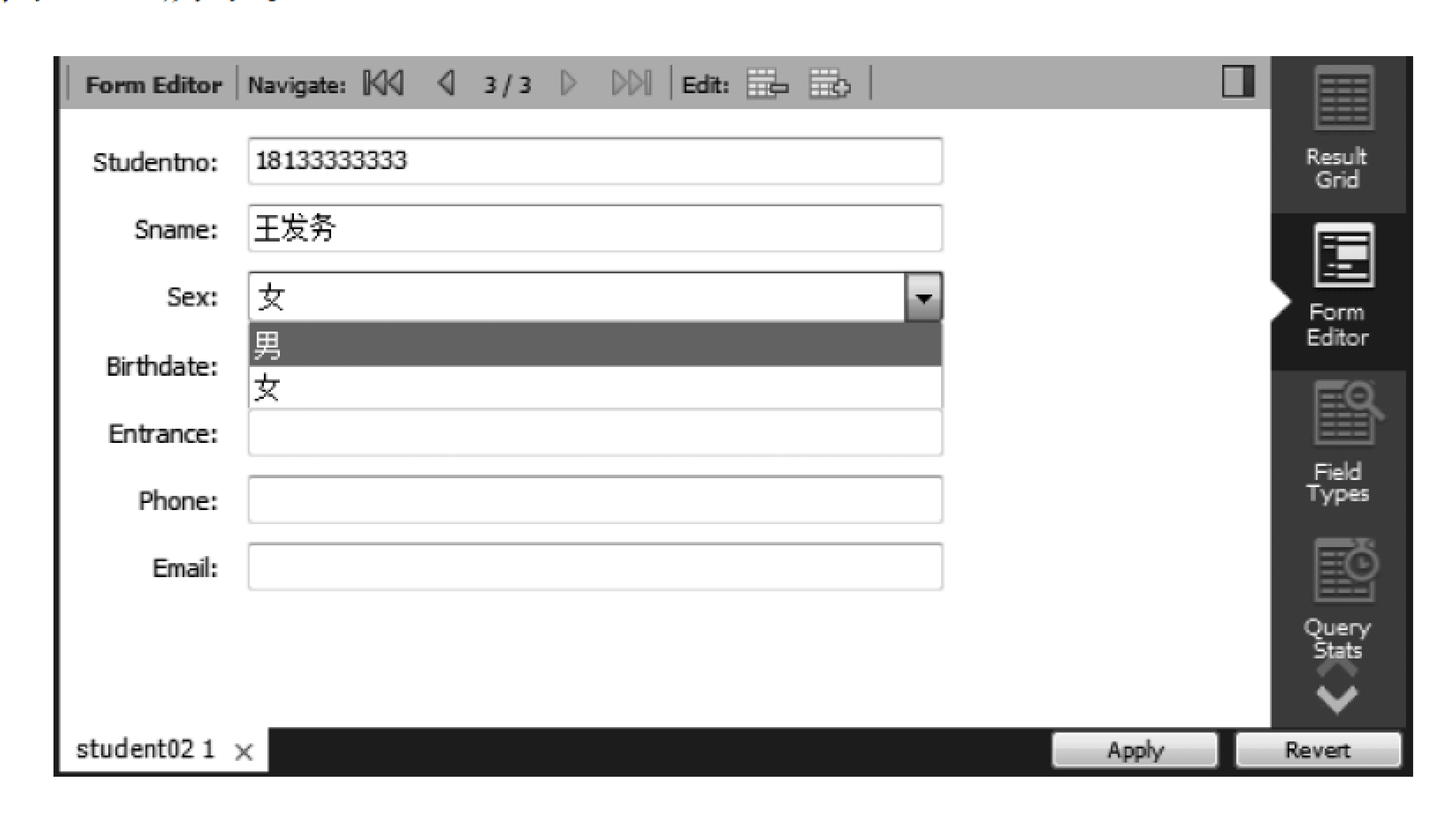


图 4-14　执行数据添加操作

(6) 同样可以在图 4-14 所示的页面中单击 Apply 按钮,对添加数据的 SQL 语句进行编辑,最后单击 Apply 按钮,完成数据的添加操作。

应用可视化工具 MySQL Workbench 可以对数据表进行利用命令实现添加、删除、修改和查询等各种操作,在学习过程中可以通过不断的练习,进行灵活的应用。

4.4 表的数据完整性

在定义表结构的同时，还可以定义与该表相关的完整性约束条件，包括实体完整性、参照完整性和用户自定义完整性。这些完整性约束条件都被存入系统的数据字典中，当用户操作表中的数据时，由数据库管理系统自动检查该操作是否违背这些完整性约束条件。如果完整性约束条件涉及该表的多个属性列，则必须定义在表级上，其他情况则既可以定义在列级上也可以定义在表级上。这些约束条件主要包括 not null(非空约束)、primary key(主键约束)、unique(唯一性约束)、foreign key(外键参照完整性约束)以及 check(检查约束)。学习创建和修改约束的方法，掌握数据约束条件的实际应用，对实现数据完整性起到不可或缺的作用。

4.4.1 非空约束

在前面的数据表定义过程中，每个字段都要有一个是否为 null 值的选择，这就是对数据表中将来的数据提出的约束条件。

(1) null(允许空值)：表示数值未确定，并不是数字"0"或字符"空格"。对于表中数据来说比较两个空值或空值与其他任何类型值比较的结果均为空值。

(2) not null(不允许空值)：表示数据列中不允许空值出现。这样可以确保数据列中必须包含有意义的值。如果数据列中设置了"不允许空值"，在向表中输入数据时，就必须输入一个值，否则该行数据将不会被收入表中。

例如，学生选课时，学号、课程号就不能为空值，因为这必须是确定值，才能描述哪位同学选的什么课。如果存在成绩字段，则成绩字段就应该允许空值，因为此时还没有结束课程，成绩是不确定的。设置表的非空约束是指在创建表时为表的某些特殊字段加上 not null 约束条件。非空约束将保证所有记录中该字段都有值。如果用户新插入的记录中，该字段为空值，则数据库系统会自动报错。

4.4.2 主键约束


主键约束

设置主键的目的主要是可以帮助 MySQL 以最快的速度查找到表中的指定信息。primary key 可指定一个字段作为表主键，也可以指定两个及以上的字段作为复合主键，其值能唯一地标识表中的每一行记录，而且 primary key 约束中的列不允许取空值。由于 primary key 约束能确保数据唯一，所以经常用来定义标志列。

可以在创建表时创建主键，也可以对表中已有主键进行修改或者增加新的主键。设置主键通常有两种方式：表的完整性约束和列的完整性约束。

1. 创建表时定义完整性约束

前面第 4.1.2 节中定义 student、course、score、teacher 等数据表时，都是采用表级约束的方式。此时，需要在语句最后加上一条 primary key(col_name,…)语句即可。若在列定义时加上关键字 primary key，就可以定义列的完整性约束主键。

【例 4-24】 创建表 course01，用列的完整性约束设置主键。

程序代码如下：

```
mysql > create table if not exists course01
      (courseno char(6) not null primary key,
       cname char(6) not null,
       type char(8) not null,
       period int(2) not null,
       exp int(2) not null,
       term int(2) not null
      );
```

说明：

(1) 主键可以是单一字段，可以是表级约束，也可以是列级约束。

(2) 当表中的主键为复合主键时，只能定义为表的完整性约束。例如，前面定义 score 表时，就是采用的这种方式。

2. 修改表的主键

修改表的主键，可以单独进行，也可以通过修改表结构来实现。

【例 4-25】 修改表 student02 的主键，删除原来主键 sname，增加 studentno 为主键。

代码和运行结果如下：

```
mysql > alter table student02 add primary key (sname);
        Query OK, 0 rows affected (0.48 sec)
        Records: 0 Duplicates: 0 Warnings: 0
mysql > alter table student02 drop primary key;
        Query OK, 3 rows affected (0.44 sec)
        Records: 3 Duplicates: 0 Warnings: 0
mysql > alter table student02 add primary key (studentno);
        Query OK, 0 rows affected (0.30 sec)
        Records: 0 Duplicates: 0 Warnings: 0
```

4.4.3 外键约束

1. 理解参照完整性

在关系型数据库中，有很多规则是和表之间的关系有关的，表与表之间往往存在一种“父子”关系。例如，字段 studentno 是一个表 score 的属性，且依赖于表 student 的主键 studentno。那么，称表 student 为父表，表 score 为子表。通常将 studentno 设为表 score 的外键，参照表 student 的主键字段通过 studentno 字段将父表 student 和子表 score 建立关联关系。



外键约束

外键的作用是建立子表与其父表的关联关系，保证子表与父表关联的数据一致性。父表中更新或删除某条信息时，子表中与之对应的信息也必须有相应的改变。

设置外键的原则：必须依赖于数据库中已存在的父表的主键；外键可以为空值。

这样，当需要在 score 表中添加、删除、修改 studentno 字段的数据时，其结果中的 studentno 值必须在 student 表中存在。即编辑成绩表 score 中的学号 studentno 时，该学号必须是 student 表存在的学号。这种类型的关系就是参照完整性约束(Referential Integrity Constraint)。

外键声明和参照完整性定义的语法格式如下：

```
constraint foreign_key_name foreign key (col_name1 [,col_name2 … . ])
references table_name(col_name1[,col_name2...)])
[on delete {restrict | cascade | set null | no action}]
[on update {restrict | cascade | set null | no action}]
```

说明：

(1) constraint foreign_key_name：定义外键约束和约束名。foreign key (col_name1 [,col_name2....])，外键引用的字段表。

(2) 外键被定义为表的完整性约束，reference_definition 中包含了外键所参照的表和列，还可以声明参照动作。

(3) restrict：当要删除或更新父表中被参照列上在外键中出现的值时，拒绝对父表的删除或更新操作。

(4) cascade：从父表删除或更新行时自动删除或更新子表中匹配的行。

(5) set null：当从父表删除或更新行时，设置子表中与之对应的外键列为 null。如果外键列没有指定 not null 限定词，这就是合法的。

(6) no action：no action 意味着不采取动作，就是如果有一个相关的外键值在被参考的表里，删除或更新父表中主要键值的企图不被允许，与 restrict 一样。

(7) set default：作用与 set null 一样，只不过 set default 是指定子表中的外键列为默认值。

2. 对已有的表添加外键

【例 4-26】 用 alter table 语句在数据库 teaching 中为表 score 添加外键约束。

代码和运行结果如下：

```
mysql > alter table score
        -> add constraint fk_st_score
        -> foreign key(studentno) references student(studentno);
    Query OK, 0 rows affected (0.33 sec)
    Records: 0 Duplicates: 0 Warnings: 0
mysql > alter table score
     -> add constraint fk_cou_score
     -> foreign key(courseno) references course(courseno);
    Query OK, 0 rows affected (0.49 sec)
    Records: 0 Duplicates: 0 Warnings: 0
```

3. 在创建表时创建外键

【例 4-27】 在 mysqltest 数据库中参照 score 表创建 score1 表，其中 studentno 作为外键，参照 student02 表中的 studentno 字段。

代码和运行结果如下：

```
mysql > create table if not exists score1
    -> (studentno char(11) not null,
    -> courseno char(6) not null,
    -> daily float(3,1) default 0,
    -> final float(3,1) default 0,
    -> primary key (studentno , courseno),
    -> foreign key(studentno)
```

```
    -> references student02(studentno)
    -> on update cascade
    -> on delete cascade);
   Query OK, 0 rows affected (0.21 sec)
```

4.4.4 检查约束


检查约束

利用主键和外键约束可以实现一些常见的完整性操作。在进行数据完整性管理时，还需要一些针对数据表的列进行限制数值范围的约束。例如，score 表中 final 字段的数值范围应为 0～100，表中 birthdate 必须大于 1990 年 12 月 31 日。这样的规则可以使用 check 完整性约束来指定。

check 约束在创建表时定义，可以定义为列完整性约束，也可以定义为表完整性约束。定义 check 约束时的格式比较简单。

【例 4-28】 在 mysqltest 数据库中，对 student02 表的 birthdate 列添加 check 约束，要求出生日期必须大于 1999 年 12 月 31 日。

代码和运行结果如下：

```
mysql> alter table student02
    -> add constraint ch_stu_birth
    -> check(birthdate>'1999-12-31');
   Query OK, 0 rows affected (0.03 sec)
   Records: 0 Duplicates: 0 Warnings: 0
```

4.4.5 唯一性约束


唯一性约束

唯一性是指所有记录中该字段的值不能重复出现。设置表的唯一性约束是指在创建表时为表的某些特殊字段加上 unique 约束条件。唯一性约束将保证所有记录中该字段的值不能重复出现。创建表时可以设置列的唯一约束，也可以在已经创建的表的列上添加唯一。例如，对 Email 字段加上唯一性约束，记录中 Email 字段上就不能出现相同的值了。

【例 4-29】 在 mysqltest 数据库中，对 student02 表的 Email 列添加唯一约束。

代码和运行结果如下：

```
mysql> alter table student02 add unique (Email);
       Query OK, 0 rows affected (0.42 sec)
       Records: 0 Duplicates: 0 Warnings: 0
mysql> select studentno,sname,email from student02;
      +------------+---------+------------+
      | studentno  | sname   | email      |
      +------------+---------+------------+
      | 16101111111 | 张思睿  | z3@162.com |
      | 16302222222 | 李佛    | 14@127.com |
      | 18133333333 | 王发务  | w5@127.com |
      +------------+---------+------------+
      3 rows in set (0.00 sec)
mysql> insert into student02
        values ('16104444444', '徐赛克', '男',
```

```
'2001-01-01', '809', '13102020207', 'z3@162.com');
ERROR 1062 (23000): Duplicate entry 'z3@162.com' for key 'Email'
                                                   //插入失败,违反唯一性约束的条件
```

说明:

(1) 一个数据表只能创建一个主键,但可以有若干个 unique 约束。

(2) 主键列值不允许为 null,而 unique 字段的值可取 null,但是必须使用 null 或 not null 声明。

(3) 一般在创建 primary key 约束时,系统会自动产生 primary key 索引。创建 unique 约束时,系统自动产生 unique 索引。

4.5 小　　结

本章介绍了利用 MySQL 创建表、查看表结构、修改表和删除表的方法。删除表的时候一定要特别小心,因为删除表的同时会删除表中的所有记录。

表是数据库存储数据的基本单位。一个表包含若干个字段或记录。数据操作是数据库管理中最基本、最重要的操作。完整性约束是保证数据正确性的重要方法。学习本章后要掌握如下重要内容:

- 创建表的方法。
- 表的完整性约束条件。
- 查看表结构的方法。
- 修改表的方法。
- 删除表的方法。
- 可视化工具 MySQL Workbench 的使用。

习　题　4

1. 选择题

(1) 要快速完全清空一张表中的记录可使用如下语句________。

A. truncate table　　B. delete table　　C. drop table　　D. clear table

(2) 使用 insert 命令插入记录时,使用________关键字会忽略导致重复关键字的错误记录。

A. no same　　B. ignore　　C. repeat　　D. unique

(3) SQL 语句中修改表结构的命令是________。

A. modify table　　B. modify structure

C. alter table　　D. alter structure

(4) 只修改列的数据类型的指令是________。

A. alter table…alter column　　B. alter table…modify column…

C. alter table…update…　　D. alter table…update column…

(5) 删除列的命令是________。

A. alter table...delete...　　B. alter table...delete column...

C. alter table...drop...　　D. alter table...drop column...

(6) 创建表时,不允许某列为空可以使用________命令。

A. not null　　B. no null　　C. not blank　　D. no blank

(7) 以下________指令无法增加记录。

A. insert into...values...　　B. insert into...select...

C. insert into...set...　　D. insert into...update...

(8) 关于 truncate table 描述不正确的是________。

A. truncate 将删除表中的所有数据

B. 表中包含 auto_increment 列,使用 truncate table 可以重置序列值为该列初始值

C. truncate 操作比 delete 操作占用资源多

D. truncate table 删除表,然后重新构建表

(9) 在创建表时,可以使用________关键字使当前建立的表为临时表。

A. ignore　　B. temporary　　C. temptable　　D. truncate

(10) 下列描述正确的是________。

A. 一个数据库只能包含一个数据表　　B. 一个数据库可以包含多个数据表

C. 一个数据库只能包含两个数据表　　D. 一个数据表可以包含多个数据库

2. 简答题

(1) 简述在创建表结构时,常用哪些数据类型？其主要功能是什么？

(2) 简述创建表时各类约束对表中数据的作用。

(3) MySQL 支持的数据完整性有哪几类？各有什么作用？

(4) 简述在 MySQL Workbench 中创建含有主键的表的步骤。

(5) 简述在 MySQL Workbench 中修改表数据的步骤。

3. 上机练习题

(1) 创建 booksmgt 数据库,在 booksmgt 数据库中使用 MySQL 语句创建表 book 和表 author,结构如下：

```
book(bookid  char(6),bookname  varchar(30),  price float(5,2))
author( authorid char(6), authorname  varchar(10), bookid  char(6), phone varchar(15))
```

设置 book 中的 bookid 为主键,author 表中的 bookid 为外键。

(2) 在 booksmgt 数据库中利用 MySQL 语句创建一个图书销售表 booksales 结构如下：

```
booksales(bookid nchar(6), sellnum int, selldate datetime)
```

分别利用 insert、delete、update 语句添加、删除和更新数据。

(3) 利用 MySQL 语句先删除表 booksales 中销售时间在 2016 年 12 月以前的记录。再删除全部记录,然后删除该表。

第5章 数据检索

数据检索是指从数据库中按照预定条件查询数据，及引用相关数据进行计算而获取所需信息的过程。查询数据是数据库操作中最常用、最重要的操作。MySQL 是通过 select 语句查询实现数据检索的。

本章将介绍利用 select 语句进行单表查询、多表连接和子查询的详细操作。

5.1 基本查询语句

select 语句是 SQL 语言从数据库中获取信息的一个基本语句。该语句可以实现从一个或多个数据库中的一个或多个表中查询信息，并将结果显示为另外一个二维表的形式，称为结果集(result set)。



select 语句

select 语句的基本语法格式可归纳如下：

```
select [all|distinct]selection_list
from table_source
[where search_condition]
[group by grouping_columns][with rollup]
[having search_condition]
[order by order_expression [asc|desc]]
[limit count]
```

说明：

- []：表示可选项格式。
- select：描述结果集的列，是一个用逗号分隔的表达式列表。每个选择列表表达式通常是对从中获取数据源列的引用，但也可能是其他表达式。all 是默认值，代表所有行。distinct 取消结果集中的重复行。
- from：指定所要查询数据源，如表、视图、表达式等。可以指定两个以上的表，表与表之间用逗号隔开。
- where：定义源表中的行要满足 select 语句的要求所必须达到的条件。
- group by：用于对查询结构根据 grouping_columns 的值进行分组。使用带 rollup 操作符的 group by 子句，指定在结果集内不仅包含由 group by 提供的正常行，还包含汇总行。
- having：having 子句是应用于分组结果集的附加条件。having 子句通常与 group by 子句一起使用，用来在 group by 子句后选择行。

- order by：用于对查询结果进行排序。
- asc|desc：用于指定行的排序，asc 代表升序，是默认值，desc 代表降序。
- limit：限制查询的输出结果行。通常与 order by 子句一起使用。

下面先介绍 select 语句的简单应用。

(1) 使用 select 语句查询一个数据表。使用 select 语句时，首先要确定所要查询的列。“*”代表所有的列。

【例 5-1】 查询 teaching 数据库 course 表中的所有数据。

代码和运行结果如下：

```
mysql> use teaching;
       Database changed
mysql> select * from course;
       +---------+----------+------+--------+-----+------+
       | courseno | cname   | type | period | exp | term |
       +---------+----------+------+--------+-----+------+
       | c05103  | 电子技术 | 必修 |     64 | 16  |    2 |
       | c05109  | C 语言   | 必修 |     48 | 16  |    2 |
       | c05127  | 数据结构 | 必修 |     64 | 16  |    2 |
       | c05138  | 软件工程 | 选修 |     48 | 8   |    5 |
       | c06108  | 机械制图 | 必修 |     60 | 8   |    2 |
       | c06127  | 机械设计 | 必修 |     64 | 8   |    3 |
       | c06172  | 铸造工艺 | 选修 |     42 | 16  |    6 |
       | c08106  | 经济法   | 必修 |     48 | 0   |    7 |
       | c08123  | 金融学   | 必修 |     40 | 0   |    5 |
       | c08171  | 会计软件 | 选修 |     32 | 8   |    8 |
       +---------+----------+------+--------+-----+------+
       10 rows in set (0.06 sec)
```

这是查询整个表中所有列的操作，还可以针对表中的某一列或多列进行查询。

(2) 查询表中的指定列。针对表中的多列进行查询，只要在 select 后面指定要查询的列名即可，多列之间用“,”分隔。

【例 5-2】 查询 student 表中的 studentno、sname 和 phone 数据。

代码和运行结果如下：

```
mysql> select studentno,sname,phone from student;
       +-----------+------+-----------+
       | studentno | sname | phone    |
       +-----------+------+-----------+
       | 18122210009 | 许东山 | 13623456778 |
       | 18122221324 | 何白露 | 13178978999 |
       | 18125111109 | 敬横江 | 15678945623 |
       | 18125121107 | 梁一苇 | 13145678921 |
       | 18135222201 | 凌浩风 | 15978945645 |
       | 18137221508 | 赵临江 | 12367823453 |
       | 19111133071 | 崔依歌 | 15556845645 |
       | 19112100072 | 宿沧海 | 12545678998 |
       | 19112111208 | 韩山川 | 15878945612 |
       | 19122203567 | 封月明 | 13245674564 |
```

```
| 19123567897  | 赵既白 | 13175689345  |
| 19126113307  | 梅惟江 | 13245678543  |
+-----------+------+-----------+
12 rows in set (0.03 sec)
```

(3) 可以从一个或多个表中获取数据。使用 select 语句进行查询，需要确定所要查询的数据在哪个表中，或在哪些表中，在对多个表进行查询时，同样使用“，”对多个表进行分隔。进行多表查询，主要采用多表连接或子查询的方式，也可以通过 where 子句中使用连接运算来确定表之间的联系，然后根据这个条件返回查询结果。

5.2 单表查询

单表查询是指从一张表中查询所需要的数据。下面将通过 select 语句的各个子句的应用介绍在单表上进行查询的常见操作。

5.2.1 select...from 基本子句的使用

select 子句的主要功能是输出字段或表达式的值，form 子句的主要功能是指定数据源。这两个子句在进行数据库表查询时，都是必选项。下面结合 select 子句的输出项的操作介绍查询语句的基本操作。

1. 为字段取别名

利用 select 语句查询数据时，输出项一般显示创建表时定义的字段名。MySQL 可以为查询显示的每个输出字段或表达式取一个别名，以增加结果集的可读性。例如，可以用 as 关键字给字段取一个中文名。实现给 select 子句中的各项取别名，其语法格式为：

```
select 项的原名   as   别名
```

【例 5-3】 在 student 表中查询出生日期在 2001 年以后的学生的学号、姓名、电话和年龄。

分析：可以通过 as 为列或表达式更改名称，增加可读性。

代码和运行结果如下：

```
mysql > select studentno as '学号',sname as '姓名',
    -> phone as '手机号',year(now()) - year(birthdate) as '年龄'
    -> from student
    -> where   year(birthdate)> 2001;
+-----------+-------+-----------+------+
| 学号       | 姓名   | 手机号      | 年龄  |
+-----------+-------+-----------+------+
| 19112100072 | 宿沧海 | 12545678998 | 15   |
| 19122203567 | 封月明 | 13245674564 | 15   |
| 19123567897 | 赵既白 | 13175689345 | 15   |
| 19126113307 | 梅惟江 | 13245678543 | 14   |
+-----------+-------+-----------+------+
4 rows in set (0.03 sec)
```

2. 使用谓词过滤记录

如果希望一个列表没有重复值，可以利用 distinct 子句从结果集中除去重复的行。当使用 distinct 子句时，需要注意以下事项：

（1）选择列表的行集中，所有值的组合决定行的唯一性。

（2）数据检索包含任何唯一值组合的行，如果不指定 distinct 子句则将所有行返回到结果集中。

【例 5-4】 在 score 表中查询期末成绩中有高于 95 的学生的学号和课程号，并按照学号排序。

分析：不管学生有几门课的成绩高于 95，只要有一门就可以显示，利用 distinct 子句可将重复行消除。

代码和运行结果如下：

```
mysql> select distinct studentno,courseno
    -> from score
    -> where final>95
    -> order by studentno;
      +-----------+--------+
      | studentno   | courseno |
      +-----------+--------+
      | 18125111109 | c08106   |
      | 18137221508 | c08171   |
      | 19112100072 | c06108   |
      | 19122203567 | c05103   |
      | 19123567897 | c06127   |
      +-----------+--------+
      5 rows in set (0.00 sec)
```

5.2.2 使用 where 子句过滤结果集



where 子句

1. 查询符合指定条件的记录数据

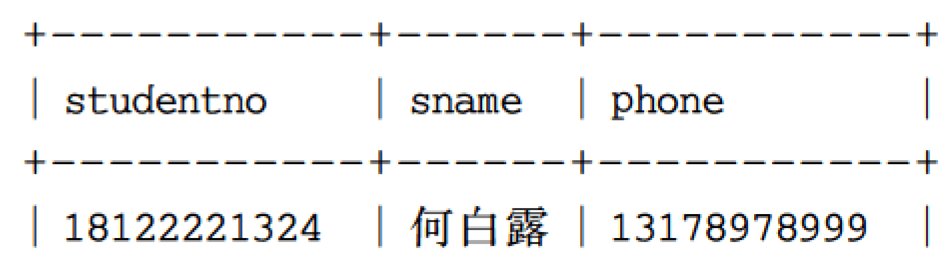
如果要从很多记录中查询出指定的记录，那么就需要一个查询的条件。设定查询条件应用的是 where 子句，通过 where 子句可以实现很多复杂的条件查询。在使用 where 子句时，需要使用一些比较运算符来确定查询的条件。

【例 5-5】 查询表 student 中入学成绩在 800 分以上的学生的学号、姓名和电话信息。

分析：本例中要求输出学号、姓名和电话信息，即为 select 子句输出表列数据源为表 student，条件为入学成绩在 800 分以上。

代码和运行结果如下：

```
mysql> select studentno,sname,phone
    -> from student
    -> where entrance>800;
```



```
      +-----------+------+-----------+
      | studentno   | sname  | phone       |
      +-----------+------+-----------+
      | 18122221324 | 何白露 | 13178978999 |
```

```
| 18135222201  | 凌浩风 | 15978945645  |
| 19122203567  | 封月明 | 13245674564  |
| 19123567897  | 赵既白 | 13175689345  |
+------------+------+------------+
4 rows in set (0.00 sec)
```

2. 带 in 关键字的查询

in 关键字可以判断某个字段的值是否在指定的集合中。如果字段的值在集合中,则满足查询条件,该记录将被查询出来;如果不在集合中,则不满足查询条件。实际上,使用 in 搜索条件相当于用 or 连接两个比较条件,如"x in(10,15)"相当于表达式"x=10 or x=15"。也可以使用 not in 关键字查询不在某取值范围内的记录行数据。

【例 5-6】 查询学号分别为 18135222201、18137221508 和 19123567897 的学生学号、课程号、平时成绩和期末成绩。

分析:检索条件中枚举某些确定值的范围,一般可以利用 in 关键字来实现。

代码和运行结果如下:

```
mysql > select studentno,courseno ,daily ,final
    -> from score
    -> where studentno in('18135222201','18137221508','19123567897');
      +------------+----------+------+-----+
      | studentno  | courseno | daily | final |
      +------------+----------+------+-----+
      | 18135222201 | c05109  | 99.0 | 92.0 |
      | 18135222201 | c08171  | 95.0 | 82.0 |
      | 18137221508 | c08106  | 80.0 | 95.0 |
      | 18137221508 | c08123  | 78.0 | 89.0 |
      | 18137221508 | c08171  | 88.0 | 98.0 |
      | 19123567897 | c05103  | 85.0 | 77.0 |
      | 19123567897 | c06127  | 99.0 | 99.0 |
      +------------+----------+------+-----+
      7 rows in set (0.07 sec)
```

3. 带 between and 的范围查询

在 where 子句中,可以使用 between 搜索条件检索指定范围内的行。使用 between 搜索条件相当于用 and 连接两个比较条件,如" x between 10 and 27"相当于表达式" x>=10 and x<=27 "。由此可见,在生成结果集中,边界值也是符合条件的。检索条件指定排除某个范围的值,一般可以利用 not between 关键字来实现。

【例 5-7】 查询选修课程号为 c05109 的学生学号和期末成绩,并且要求平时成绩在 80～95 分。

分析:检索条件设置在某个范围内,一般可以利用 between 关键字来实现。

代码和运行结果如下:

```
mysql > select studentno, final
    -> from score
    -> where courseno = 'c05109' and daily between 80 and 95;
      +------------+-------+
      | studentno  | final  |
```

```
+-----------+-------+
| 18122221324 | 77.0  |
| 18125121107 | 62.0  |
| 19112100072 | 86.0  |
| 19112111208 | 91.0  |
+-----------+-------+
4 rows in set (0.00 sec)
```

4. 带 like 的字符匹配查询

使用通配符结合的 like 搜索条件，通过进行字符串的比较来选择符合条件的行。当使用 like 搜索条件时，模式字符串中的所有字符都有意义，包括开头和结尾的空格。like 主要用于字符类型数据。字符串内的英文字母和汉字都算一个字符。也可用通配符并使用 not like 作为查询条件。

like 属于较常用的比较运算符，通过它可以实现模糊查询。它有两种通配符："%"和下画线"_"：

- "%"可以匹配一个或多个字符，可以代表任意长度的字符串，长度可以为 0。
- "_"只匹配一个字符。

【例 5-8】 在 student 表中显示所有姓何或姓韩的学生的姓名、生日和 Email。

分析：设置 where 条件实现上述要求，需要采用 or 和 like 等逻辑运算。like 操作符可以和通配符一起将列的值与某个特定的模式作比较，列的数据类型可以是任何字符串类型。

代码和运行结果如下：

```
mysql> select sname, birthdate, Email
    -> from student
    -> where sname like '何%' or sname like '韩%';
+------+-----------+-------------+
| sname | birthdate  | Email          |
+------+-----------+-------------+
| 何白露 | 2000-12-04 | heyy@sina.com |
| 韩山川 | 2001-02-14 | han@163.com   |
+------+-----------+-------------+
2 rows in set (0.03 sec)
```

5. 用 is null 关键字查询空值

涉及空值的查询用 null 来表示。create table 语句或 alter table 语句中的 null 表明在列中允许存在被称为 null 的特殊数值，它不同于数据库中的其他任何值。在 select 语句中，where 子句通常会返回比较的计算结果为真的行。

那么，在 where 子句中，如何处理 null 的值的比较呢？为了取得列中含有 null 的行，MySQL 语句包含了操作符功能 is [not] null。

说明：

- 一个字段值是空值或者不是空值，要表示为"is null"或"is not null"；不能表示为"=null"或"<>null"。
- 如果写成"字段=null"或"字段<>null"，系统的运行结果都直接处理为 null 值，按照 false 处理而不报错。

where 子句有以下通用格式：

```
column is [not] null
```

下面通过例题介绍空值查询的方法。

【例 5-9】 在 se_score 表中添加成绩字段 score，查询 se_score 表中学生的学号、课程号和成绩。

分析：学生选修课程表 se_course 中的成绩允许空值，以此是否成绩为空值作为查询条件，即可查到学生的选课情况。

代码和运行结果如下：

```
mysql > alter table se_course
    -> add score float(3,1) null aften teacherno;
       Query OK, 0 rows affected (1.27 sec)
       Records: 0 Duplicates: 0 Warnings: 0
mysql > select studentno, courseno,teacherno, score
    -> from se_course
    -> where score is null;
        +------------+----------+----------+-------+
        | studentno  | courseno | teacherno |score |
        +------------+----------+----------+-------+
        |19120000111 | co1236   | t01237   | null  |
        |19120000222 | co1237   | t01239   | null  |
        +------------+----------+----------+-------+
        2 rows in set (0.00 sec)
```

6. 带 and 的多条件查询

where 子句的主要功能是利用指定的条件选择结果集中的行。符合条件的行出现在结果集中，不符合条件的行将不出现在结果集中。利用 where 子句指定行时，条件表达式中的字符型和日期类型值要放到单引号内，数值类型的值直接出现在表达式中。

【例 5-10】 在 score 表中显示期中成绩高于 90 分、期末成绩高于 85 分的学生学号、课程号和成绩。

分析：设置 where 条件实现上述要求，需要采用 and 逻辑运算，将两个比较运算表达式连接起来。

代码和运行结果如下：

```
mysql > select studentno,courseno,daily,final
    -> from score
    -> where daily > = 90 and final > =  85;
        +-----------+----------+-------+-------+
        | studentno  | courseno | daily | final |
        +-----------+----------+-------+-------+
        | 18135222201 | c05109 | 99.0 | 92.0  |
        | 19112100072 | c06108 | 97.0 | 97.0  |
        | 19123567897 | c06127 | 99.0 | 99.0  |
        +-----------+----------+-------+-------+
        3 rows in set (0.00 sec)
```

7. 带 or 的多条件查询

带 or 的多条件查询，实际上是指只要符合多条件中的一个，记录就会被搜索出来；如果不满足这些查询条件中的任何一个，这样的记录将被排除掉。or 可以用来连接两个条件表达式。而且，可以同时使用多个 or 关键字连接多个条件表达式。

【例 5-11】 查询计算机学院的具有高级职称教师的教师号、姓名和从事的专业。

分析：where 子句设置的条件包括部门和职称，其中高级职称又包括教授和副教授两类，需要包括 or 和 and 两种逻辑运算。

代码和运行结果如下：

```
mysql > select teacherno,tname, major
    -> from teacher
    -> where department = '计算机学院' and (prof = '副教授'or prof = '教授');
       +----------+--------+--------+
       | teacherno | tname  | major  |
       +----------+--------+--------+
       | t05001   | 苏超然 | 软件工程 |
       | t05003   | 孙释安 | 网络安全 |
       | t05011   | 卢敖治 | 软件工程 |
       +----------+--------+--------+
       3 rows in set (0.06 sec)
```

5.2.3 使用 order by 子句对结果集排序



order by 子句

使用 order by 子句可以对查询的结果进行升序(asc)或降序(desc)排列。排序可以依照某个列的值，若列值相等则根据第 2 个属性的值，以此类推。

利用 order by 子句进行排序，需要注意如下事项和原则：

(1) 默认情况下，结果集按照升序排列。也可以在输出项的后面加上关键字 desc 来实现降序输出。对含有 null 值的列进行排序时，如果是按升序排列，null 值将出现在最前面，如果是按降序排列，null 值将出现在最后。

(2) order by 子句包含的列并不一定出现在选择列表中。

(3) order by 子句可以通过指定列名、函数值和表达式的值进行排序。

(4) order by 子句不可以使用 text、ntext 或 image 类型的列。

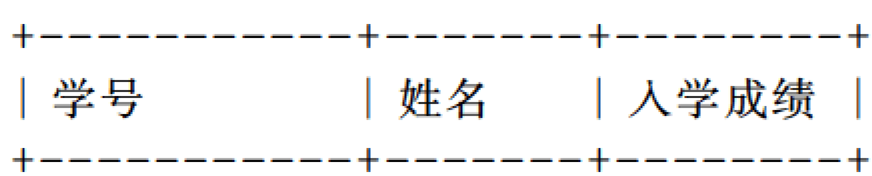
(5) 在 order by 子句中可以同时指定多个排序项。

【例 5-12】 在 student 表中查询高于 850 分的学生学号、姓名和入学成绩，并按照入学成绩的降序排列。

分析：升序 asc 是默认值，而降序 desc 必须表明，也可以给字段取别名。

代码和运行结果如下：

```
mysql > select studentno 学号,sname 姓名 ,entrance 入学成绩
    -> from student
    -> where entrance > 850
    -> order by entrance desc;
```



```
       +-----------+-------+--------+
       | 学号       | 姓名   | 入学成绩 |
       +-----------+-------+--------+
```

```
| 19123567897 | 赵既白 |      999 |
| 19122203567 | 封月明 |      898 |
| 18122221324 | 何白露 |      879 |
| 18135222201 | 凌浩风 |      867 |
+-----------+-------+--------+
4 rows in set (0.02 sec)
```

【例 5-13】 在 score 表中查询总评成绩大于 90 分的学生的学号、课程号和总评成绩，并先按照课程号的升序、再按照总评成绩的降序排列。总评成绩计算公式如下：

```
总评成绩 = daily * 0.2 + final * 0.8
```

分析：本例利用表达式作比较和排序的依据。

代码和运行结果如下：

```
mysql> select courseno 课程号,daily * 0.2 + final * 0.8 as '总评',studentno 学号
    -> from score
    -> where daily * 0.2 + final * 0.8 > 90
    -> order by courseno, daily * 0.2 + final * 0.8 desc;
+--------+-----+------------+
| 课程号  | 总评 | 学号         |
+--------+-----+------------+
| c05103 | 91.4 | 19122203567 |
| c05109 | 93.4 | 18135222201 |
| c06108 | 97.0 | 19112100072 |
| c06108 | 93.8 | 19112111208 |
| c06127 | 99.0 | 19123567897 |
| c08106 | 95.0 | 18125111109 |
| c08106 | 92.0 | 18137221508 |
| c08123 | 90.6 | 18125111109 |
| c08171 | 96.0 | 18137221508 |
+--------+-----+------------+
9 rows in set (0.00 sec)
```

5.2.4 group by 子句和 having 子句的使用

group by 子句可以将查询结果按属性列或属性列组合在行的方向上进行分组，每组在属性列或属性列组合上具有相同的聚合值。如果聚合函数没有使用 group by 子句，则只为 select 语句报告一个聚合值。


group by 子句

将一列或多列定义成为一组，使组内所有的行在那些列中的数值相同。出现在查询的 select 列表中的每一列都必须同时出现在 group by 子句中。

1. 使用 group by 关键字来分组

单独使用 group by 关键字，查询结果只显示每组的一条记录。

【例 5-14】 利用 group by 子句对 score 表数据分组，显示每个学生的学号和平均总评成绩。总评成绩计算公式如下：

```
总评成绩 = daily * 0.3 + final * 0.7
```

分析：通过学号分组，可以求出每个学生的平均总评成绩。avg()函数用于求平均值，

round()函数用于对平均值的某位数据进行四舍五入。

代码和运行结果如下：

```
mysql> select studentno 学号, round(avg(daily * 0.3 + final * 0.7),2) as'平均分'
    -> from score
    -> group by studentno;
+-----------+------+
| 学号       | 平均分 |
+-----------+------+
| 18122210009 | 85.15 |
| 18122221324 | 75.50 |
| 18125111109 | 90.13 |
| 18125121107 | 78.00 |
| 18135222201 | 90.00 |
| 18137221508 | 90.40 |
| 19111133071 | 76.70 |
| 19112100072 | 91.65 |
| 19112111208 | 91.20 |
| 19122111208 | 70.30 |
| 19122203567 | 87.37 |
| 19123567897 | 89.20 |
| 19126113307 | 79.45 |
+-----------+------+
13 rows in set (0.04 sec)
```

2. group by 关键字与 group_concat()函数一起使用

使用 group by 关键字和 group_concat()函数查询，可以将每个组中的所有字段值都显示出来。

【例 5-15】 使用 group by 关键字和 group_concat()函数对 score 表中的 studentno 字段进行分组查询。可以查看选学该门课程的学生学号。

代码和运行结果如下：

```
mysql> select courseno 课程号,group_concat(studentno) 选课学生学号
    -> from score
    -> group by courseno ;
+----------+----------------------------------------------+
| 课程号    |选课学生学号                                    |
+----------+----------------------------------------------+
| c05103   | 18122210009,18122221324,18125121107, ……       |
| c05108   | 19122203567                                   |
| c05109   | 18122210009,18122221324,18125121107, ……       |
| c06108   | 19112100072,19112111208,19126113307           |
| c06127   | 19122111208,19122203567,19123567897           |
| c08106   | 18125111109,18137221508                       |
| c08123   | 18125111109,18137221508                       |
| c08171   | 18125111109,18135222201,18137221508,19126113307 |
+----------+----------------------------------------------+
8 rows in set (0.00 sec)
```

3. group by 关键与 having 一起使用

select 语句中的 where 和 having 子句控制用数据源表中的哪些行来构造结果集。where 和 having 是筛选，这两个子句指定一系列搜索条件，只有那些满足搜索条件的行才用来构造结果集。

having 子句通常与 group by 子句结合使用，尽管指定该子句时也可以不带 group by。having 子句指定在应用 where 子句的筛选后要进一步应用的筛选。

【例 5-16】 查询选课在 3 门以上且各门课程期末成绩均高于 75 分的学生的学号及其总成绩，查询结果按总成绩降序列出。

分析：可以利用 having 子句筛选分组结果，使之满足 count(*)>=3 的条件即可。

代码和运行结果如下：

```
mysql> select studentno 学号,sum(daily * 0.3 + final * 0.7) as '总分'
    -> from score
    -> where final >= 75
    -> group by studentno
    -> having count( * )>= 3
    -> order by sum(daily * 0.3 + final * 0.7) desc;
      +------------+-------+
      |学号         | 总分   |
      +------------+-------+
      | 18137221508 | 271.2 |
      | 18125111109 | 270.4 |
      | 19122203567 | 262.1 |
      +------------+-------+
      3 rows in set (0.00 sec)
```

5.2.5 用 limit 限制查询结果的数量

limit 是用来限制查询结果的数量的子句。可以指定查询结果从哪条记录开始显示。还可以指定一共显示多少条记录。limit 可以指定初始位置，也可以不指定初始位置。



limit 子句

【例 5-17】 查询 student 表的学号、姓名、出生日期和电话，按照 entrance 进行降序排列，显示前 3 条记录。

代码和运行结果如下：

```
mysql> select studentno,sname,birthdate,phone
    -> from student
    -> order by entrance desc
    -> limit 3;
      +------------+--------+------------+------------+
      | studentno   | sname  | birthdate  | phone       |
      +------------+--------+------------+------------+
      | 19123567897 | 赵既白 | 2002-08-04 | 13175689345 |
      | 19122203567 | 封月明 | 2002-09-09 | 13245674564 |
      | 18122221324 | 何白露 | 2000-12-04 | 13178978999 |
      +------------+--------+------------+------------+
      3 rows in set (0.00 sec)
```

使用 limit 还可以从查询结果的中间部分取值。首先要定义两个参数,参数 1 是开始读取的第 1 条记录的编号(注意在总查询结果中,第 1 条记录编号为 0);参数 2 是要查询记录的个数。

【例 5-18】 查询 score 表中,期末成绩 final 高于 85 分的,按照平时成绩 daily 进行升序排列,从编号 2 开始,查询 5 条记录。

代码和运行结果如下:

```
mysql> select * from score
    -> where final>85
    -> order by daily asc
    -> limit 2,5;
      +-------------+---------+------+------+
      | studentno   | courseno | daily | final |
      +-------------+---------+------+------+
      | 18122210009 | c05109  | 77.0 | 91.0 |
      | 18125111109 | c08171  | 77.0 | 92.0 |
      | 18137221508 | c08123  | 78.0 | 89.0 |
      | 18125111109 | c08106  | 79.0 | 99.0 |
      | 19122203567 | c06127  | 79.0 | 88.0 |
      +-------------+---------+------+------+
      5 rows in set (0.00 sec)
```

5.3 聚合函数查询

MySQL 的常用聚合函数包括 count()、sum()、avg()、max()和 min()等。其中,count()用来统计记录的条数;sum()用来计算字段的值的总和;avg()用来计算字段的值的平均值;max()用来查询字段的最大值;min()用来查询字段的最小值。利用聚合函数可以满足表中记录的聚合运算。例如,需要计算学生成绩表中的平均成绩,可以使用 avg()函数。group by 关键字通常需要与聚合函数一起使用。



聚合函数

5.3.1 count()函数

count()函数对于除"*"以外的任何参数,返回所选择聚合中非 null 值的行的数目;对于参数"*",返回选择聚合所有行的数目,包含 null 值的行。没有 where 子句的 count(*)是经过内部优化的,能够快速地返回表中所有的记录总数。

【例 5-19】 通过查询求 18 级学生的总数。

分析:求学生数即为求符合要求的记录行数,一般利用 count()函数实现。

代码和运行结果如下:

```
mysql> select count(studentno) as '18 级学生数'
    -> from student
    -> where substring(studentno,1,2) = '18';
      +-----------+
      | 18 级学生数 |
      +-----------+
```

```
|          6   |
+-----------+
1 row in set (0.03 sec)
```

5.3.2 sum()函数和 avg()函数

sum()函数可以求出表中某个字段取值的总和。avg()函数可以求出表中某个字段取值的平均值。

【例 5-20】 查询 score 表中学生的期末总成绩大于 270 分的学生学号、总成绩及平均成绩。

分析：先按照 studentno 对 final 值进行分组，再利用 sum()函数和 avg()函数分别求期末总成绩和平均值，然后进行期末总成绩大于 270 分学生的筛选。

代码和运行结果如下：

```
mysql> select studentno 学号, sum(final) 总分, avg(final) 平均分
    -> from score
    -> group by studentno
    -> having sum(final)> 270
    -> order by studentno;
      +-----------+-------+--------+
      |学号        | 总分   | 平均分   |
      +-----------+-------+--------+
      | 18125111109 | 283.0 | 94.33333 |
      | 18137221508 | 282.0 | 94.00000 |
      | 19122203567 | 275.0 | 91.66667 |
      +-----------+-------+--------+
      3 rows in set (0.00 sec)
```

5.3.3 max()函数和 min()函数

max()函数可以求出表中某个字段取值的最大值，min()函数可以求出表中某个字段取值的最小值。

【例 5-21】 查询选修课程号为 c05109 的课程的期末最高分、最低分及之间相差的分数。

分析：分别利用 max()和 min()函数求得 final 的最大最小值。

代码和运行结果如下：

```
mysql> select max(final) 最高分, min(final) 最低分,
    -> max(final) - min(final) as 分差
    -> from score
    -> where (courseno = 'c05109');
      +------+------+----+
      | 最高分 | 最低分 | 分差 |
      +------+------+----+
      |  92.0 |  62.0 | 30.0 |
      +------+------+----+
      1 row in set (0.02 sec)
```

5.3.4 利用 group by 子句与 with rollup 一起进行统计

MySQL 中 with rollup 的应用,可以在分组统计数据的基础上再进行相同的总体统计。例如,对于成绩表中,查询某一门课的平均值和所有成绩的平均值,普通的 group by 语句是不能实现的。

【例 5-22】 查询 score 表中每一门课的期末平均值和所有成绩的平均值。

分析:如果使用有 with rollup 子句的 group by 语句,则可以实现这个要求。

代码和运行结果如下:

```
mysql> select courseno 课程号,avg(final) 课程期末平均分
    -> from score
    -> group by courseno with rollup;
    +--------+------------+
    | 课程号  | 课程期末平均分|
    +--------+------------+
    | c05103 |   79.83333 |
    | c05108 |   89.00000 |
    | c05109 |   83.00000 |
    | c06108 |   91.33333 |
    | c06127 |   84.66667 |
    | c08106 |   97.00000 |
    | c08123 |   90.50000 |
    | c08171 |   87.75000 |
    | null   |   85.82143 |
    +--------+------------+
    9 rows in set (0.00 sec)
```

运行结果中,最后一行即为所有成绩的平均分。

5.4 多表连接

连接是关系型数据库中常用的多表查询数据的模式,连接可以根据各个表之间的逻辑关系来利用一个表中的数据选择另外的表中的行实现数据的关联操作。要在数据库中完成复杂的查询,必须将两个或两个以上的表连接起来。连接条件可在 from 或 where 子句中指定。连接条件与 where 和 having 搜索条件组合,用于控制 from 子句引用的数据源中所选定的行。

MySQL 处理连接时,查询引擎从多种可能的方法中选择最高效的方法处理连接。尽管不同连接的物理执行可以采用多种不同的优化,但逻辑序列都是通过应用 from、where 和 having 子句中的连接条件和搜索条件实现的。

连接条件中用到的字段虽然不必具有相同的名称或相同的数据类型,但是如果数据类型不相同,则必须兼容或可进行隐性转换。

MySQL 显式定义了连接操作,增强了查询的可读性。被显式定义的与连接有关的关键字如下:

(1) Inner Join:内连接,结果只包含满足条件的列。

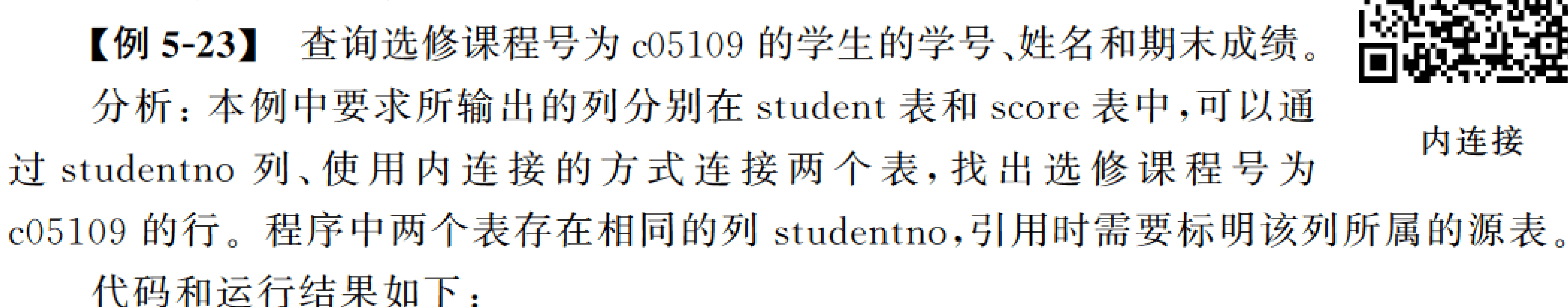
(2) Left Outer Join：左外连接，结果包含满足条件的行及左侧表中的全部行。

(3) Right Outer Join：右外连接，结果包含满足条件的行及右侧表中的全部行。

(4) Cross Join：结果只包含两个表中所有行的组合，指明两表间的笛卡儿操作。

5.4.1 内连接

内连接(Inner Join)查询是通过比较数据源表间共享列的值，从多个源表检索符合条件的行的操作。可以使用等号运算符的连接，也可以连接两个不相等的列中的值。



内连接

【例 5-23】 查询选修课程号为 c05109 的学生的学号、姓名和期末成绩。

分析：本例中要求所输出的列分别在 student 表和 score 表中，可以通过 studentno 列、使用内连接的方式连接两个表，找出选修课程号为 c05109 的行。程序中两个表存在相同的列 studentno，引用时需要标明该列所属的源表。

代码和运行结果如下：

```
mysql > select student.studentno,sname,final
    -> from student inner join score
    -> on student.studentno =  score.studentno
    -> where score.courseno  =  'c05109';
       +-----------+------+------+
       | studentno  | sname | final  |
       +-----------+------+------+
       | 18122210009 | 许东山 | 91.0   |
       | 18122221324 | 何白露 | 77.0   |
       | 18125121107 | 梁一苇 | 62.0   |
       | 18135222201 | 凌浩风 | 92.0   |
       | 19111133071 | 崔依歌 | 82.0   |
       | 19112100072 | 宿沧海 | 86.0   |
       | 19112111208 | 韩山川 | 91.0   |
       +-----------+------+------+
       7 rows in set (0.01 sec)
```

还有一种方法，就是直接通过 where 子句的复合条件查询，可以实现与内连接的同样结果。代码如下：

```
mysql > select student.studentno,sname,final
    -> from student,score
    -> where student.studentno =  score.studentno
    -> and score.courseno  =  'c05109';
```

5.4.2 外连接

外连接(Outer Join)包括满足搜索条件的连接表中的所有行，甚至包括在其他连接表中没有匹配行的一个表中的行。对于当一个表中的行与其他表中的行不匹配时返回的结果集行，为解析为不存在相应行的表的所有结果集列提供 null 值。



外连接

外连接会返回 from 子句中提到的至少一个表或视图中的所有行，只要这些行符合任何 where 或 having 搜索条件。将检索通过左外部连接引

用的左表中的所有行，以及通过右外部连接引用的右表中的所有行。

外连接是使用 outer join 关键字将两个表连接起来。外连接生成的结果集不仅包含符合连接条件的行数据，而且还包括左表(左外连接时的表)、右表(右外连接时的表)中所有的数据行。

1. 左外连接

左外连接(Left Outer Join)是指将左表中的所有数据分别与右表中的每条数据进行连接组合，返回的结果除内连接的数据外，还包括左表中不符合条件的数据，并在右表的相应列中添加 null 值。

【例 5-24】 在 mysqltest 数据库中利用左外连接方式查询学生的学号、姓名、平时成绩和期末成绩。

分析：当右表中的行与左表中的行不匹配时，左外连接方式将会将右表的所有结果集列赋予 null 值。

代码和运行结果如下：

```
mysql > use mysqltest;
        Database changed
mysql > select student02.studentno,sname,daily,final
    -> from student02 left join score1
    -> on student02.studentno = score1.studentno;
      +-----------+-------+------+-----+
      | studentno | sname | daily | final |
      +-----------+-------+------+-----+
      | 16101111111 | 张思睿 | null | null |
      | 16302222222 | 李佛 | null | null |
      | 18133333333 | 王发务 | null | null |
      +-----------+-------+------+-----+
      3 rows in set (0.09 sec)
```

2. 右外连接

右外连接(Right Outer Join)也是外部连接的一种，其中包含 join 子句中最右侧表的所有行。如果右侧表中的行与左侧表中的行不匹配，将为结果集中来自左侧表的所有列分配 null 值。

【例 5-25】 利用右外连接方式查询教师的排课情况。

分析：当左表中的行与右表中的行不匹配时，右外连接方式将会将左表的所有结果集列赋予 null 值。

代码和运行结果如下：

```
mysql > select teacher.teacherno,tname, major, courseno
    -> from teacher right join teach_course
    -> on teacher.teacherno  =  teach_course.teacherno;
      +---------+------+--------+--------+
      | teacherno | tname | major | courseno |
      +---------+------+--------+--------+
      | t05001 | 苏超然 | 软件工程 | c05109 |
      | t05002 | 常杉 | 会计学 | c05127 |
      | t05003 | 孙释安 | 网络安全 | c05127 |
      | t05011 | 卢敖治 | 软件工程 | c05138 |
```

```
| t05017      | 茅佳峰 | 软件测试 | c05127   |
| t06011      | 夏南望 | 机械制造 | c06127   |
| t06023      | 葛庭宇 | 铸造工艺 | c06172   |
| t07019      | 韩既乐 | 经济管理 | c08123   |
| t08017      | 时观   | 金融管理 | c08106   |
| null        | null   | null     | c09091   |
+-----------+------+--------+--------+
10 rows in set (0.00 sec)
```

5.4.3 交叉连接


交叉连接

交叉连接(Cross Join)是在没有 where 子句的情况下,产生的表的笛卡儿积。两个表作交叉连接时,结果集大小为二者行数之积。该种方式在实际过程中用得很少。

【例 5-26】 显示 student 表和 score 表的笛卡儿积。

分析:其结果集 336 行数据,应是 student 表数据行数与 score 表行数的乘积数。

代码和运行结果如下:

```
mysql > select student.studentno,sname,score.*
    -> from student cross join score;
+-------------+-------+------------+---------+-------+------+
| studentno   | sname | studentno  | courseno | daily | final |
+-------------+-------+------------+---------+-------+------+
| 18122210009 | 许东山 | 18122210009 | c05103  | 87.0 | 82.0 |
| 18122221324 | 何白露 | 18122210009 | c05103  | 87.0 | 82.0 |
…
| 19122203567 | 封月明 | 19126113307 | c08171  | 88.0 | 79.0 |
| 19123567897 | 赵既白 | 19126113307 | c08171  | 88.0 | 79.0 |
| 19126113307 | 梅惟江 | 19126113307 | c08171  | 88.0 | 79.0 |
+-------------+-------+------------+---------+-------+------+
336 rows in set (0.01 sec)
```

5.4.4 连接多个表


连接多个表

从理论上说,对于使用 select 语句进行连接的表数目没有上限。但在一条 select 语句中连接的表多于 10 个,那么数据库就很可能达不到最优化设计,MySQL 引擎的执行计划会变得非常烦琐。

需要注意的是,对于 3 个以上关系表的连接查询,一般遵循下列规则:连接 n 个表至少需要 n−1 个连接条件,以避免笛卡儿积的出现。为了缩小结果集,采用多于 n−1 个连接条件或使用其他条件都是允许的。

【例 5-27】 查询 18 级学生的学号、姓名、课程名、期末成绩及学分。

分析:本例要求输出的各项分别存在于 student、course 和 score 三个表中,因此至少需要创建两个连接条件。每 16 个学时,计为 1 学分。

代码和运行结果如下:

```
mysql > select student.studentno,sname,cname,final,round(period/16,1)
    -> from score join student on student.studentno = score.studentno
```

```
    ->             join course on score.courseno = course.courseno
    -> where substring(student.studentno,1,2) = '18';
+-------------+----------+-------------+----------+-------------------+
| studentno   | sname    | cname       | final    |round(period/16,1) |
+-------------+----------+-------------+----------+-------------------+
| 18122210009 | 许东山   | 电子技术    | 82.0     |               4.0 |
| 18122210009 | 许东山   | C 语言      | 91.0     |               3.0 |
| 18122221324 | 何白露   | 电子技术    | 62.0     |               4.0 |
| 18122221324 | 何白露   | C 语言      | 77.0     |               3.0 |
| 18125111109 | 敬横江   | 经济法      | 99.0     |               3.0 |
| 18125111109 | 敬横江   | 金融学      | 92.0     |               2.5 |
| 18125111109 | 敬横江   | 会计软件    | 92.0     |               2.0 |
| 18125121107 | 梁一苇   | 电子技术    | 91.0     |               4.0 |
| 18125121107 | 梁一苇   | C 语言      | 62.0     |               3.0 |
| 18135222201 | 凌浩风   | C 语言      | 92.0     |               3.0 |
| 18135222201 | 凌浩风   | 会计软件    | 82.0     |               2.0 |
| 18137221508 | 赵临江   | 经济法      | 95.0     |               3.0 |
| 18137221508 | 赵临江   | 金融学      | 89.0     |               2.5 |
| 18137221508 | 赵临江   | 会计软件    | 98.0     |               2.0 |
+-------------+----------+-------------+----------+-------------------+
14 rows in set (0.00 sec)
```

5.4.5 合并多个结果集

union 操作符可以将多个 select 语句的返回结果组合到一个结果集中。当要检索的数据在不同的结果集中,并且不能够利用一个单独的查询语句得到时,可以使用 union 合并多个结果集。将两个或更多查询的结果合并为单个结果集,该结果集包含联合查询中的所有查询的全部行。union 运算不同于使用连接合并两个表中的列的运算。

使用 union 合并两个查询结果集时,所有查询中的列数和列的顺序必须相同且数据类型必须兼容。

union 操作符基本语法格式如下:

```
select_statement union [all] select_statement
```

其中,格式中的参数说明如下:

(1) select_statement: select 语句。

(2) union: 指定组合多个结果集并返回为单个结果集。

(3) all: 将所有行合并到结果中,包括重复的行。如果不指定,将删除重复的行。

【例 5-28】 在 mysqltest 数据库中利用 student 表创建 student01,将 student01 和 student 表的部分查询结果集合并。

分析: 虽然两个表的结构不同,但需要合并的两个结果集结构和列的数据类型兼容。

代码和运行结果如下:

```
mysql> createtable student01 as
    -> select studentno, sname, phone from teaching.student;
    Query OK, 12 rows affected (0.43 sec)
    Records: 12 Duplicates: 0 Warnings: 0
```

```
mysql > select studentno, sname, phone from student01
    -> where phone like '%131%'
    -> union
    -> select studentno, sname, phone from teaching.student
    -> where phone like '%132%';
+------------+---------+-------------+
| studentno  | sname   | phone       |
+------------+---------+-------------+
| 18122221324 | 何白露 | 13178978999 |
| 18125121107 | 梁一苇 | 13145678921 |
| 19123567897 | 赵既白 | 13175689345 |
| 19122203567 | 封月明 | 13245674564 |
| 19126113307 | 梅惟江 | 13245678543 |
+------------+---------+-------------+
5 rows in set (0.02 sec)
```

5.5 子 查 询

子查询就是一个嵌套在 select、insert、update 或 delete 语句或其他子查询中的查询。部分子查询和连接可以相互替代，使用子查询也可以替代表达式。通过子查询可以把一个复杂的查询分解成一系列的逻辑步骤，利用单个语句的组合解决复杂的查询问题。

(1) 子查询的执行过程。MySQL 对嵌套查询的处理过程是从内层向外层处理，即先处理最内层的子查询，然后把查询的结果用于其外查询的查询条件，再层层向外求解，最后得出查询结果。

(2) 子查询和连接的关系。一般情况下，包含子查询的查询语句可以写成连接查询的方式。因此，通过子查询也可以实现多表之间的查询。在有些方面，多表连接的性能要优于子查询，原因是连接不需要查询优化器执行排序等额外的操作。

(3) 子查询中的常见运算。子查询中可以包括 in、not in、any、all、exists、not exists 等逻辑运算符，也可以包含比较运算符，如"＝""！＝""＞"和"＜"等。

(4) 子查询的类型。根据子查询的结果又可以将 MySQL 子查询分为 4 种类型。

- 返回一个表的子查询是表子查询。
- 返回带有一个或多个值的一行的子查询是行子查询。
- 返回一行或多行，但每行上只有一个值的是列子查询。
- 只返回一个值的是标量子查询。从定义上讲，每个标量子查询都是一个列子查询和行子查询。

(5) 使用子查询时应该注意如下的事项：

- 子查询需要用括号括起来。子查询中也可以再包含子查询，嵌套可以多至 32 层。
- 当需要返回一个值或一个值列表时，可以利用子查询代替一个表达式。也可以利用子查询返回含有多个列的结果集替代表或连接操作相同的功能。
- 子查询不能够检索数据类型为 varchar(max)、nvarchar(max)和 varbinary(max)的列。
- 子查询使用 order by 时，只能在外层使用，不能在内层使用。

5.5.1 利用子查询做表达式



select 子句中的子查询

在 MySQL 语句中,可以把子查询的结果当成一个普通的表达式来看待,用在其外查询的选择条件中。此时子查询必须返回一个值或单个列值列表,此时的子查询可以替换 where 子句中包含 in 关键字的表达式。

【例 5-29】 查询学号为 18125121107 的学生的入学成绩、所有学生的平均入学成绩及该学生成绩与所有学生的平均入学成绩的差。

分析:利用子查询求学生的平均入学成绩,作为 select 语句的输出项表达式。

代码和运行结果如下:

```
mysql> select studentno,sname,entrance ,
    -> (select avg(entrance) from student ) 平均成绩,
    -> entrance - (select avg(entrance) from student ) 分差
    -> from student
    -> where studentno = '18125121107';
      +------------+-------+---------+---------+---------+
      | studentno  | sname | entrance | 平均成绩 | 分差    |
      +------------+-------+---------+---------+---------+
      | 18125121107 | 梁一苇 |     777 | 807.0833 | -30.0833 |
      +------------+-------+---------+---------+---------+
      1 row in set (0.04 sec)
```

5.5.2 利用子查询生成派生表



from 子句中的子查询

select 的数据源由 from 子句指定,from 子句可以指定单个表或者多个表,还可以查询来自视图、临时表或结果集的数据源。即可以利用子查询生成一个派生表,用于替代 from 子句中的数据源表,派生表可以定义一个别名,即子查询的结果集可以作为外层查询的源表。实际上是在 from 子句中使用子查询作为派生表数据源。

【例 5-30】 查询期末成绩高于 85 分、总评成绩高于 90 分的学生的学号、课程号和总评成绩。

分析:利用子查询过滤出期末成绩高于 85 分的结果集,以 TT 命名,然后再对结果集 TT 中的数据进行查询。

代码和运行结果如下:

```
mysql> select TT.studentno 学号 ,TT.courseno 课程号 ,
    ->           TT.final * 0.8 + TT.daily * 0.2 总评
    -> from (select * from score where final > 85) as TT
    -> where TT.final * 0.8 + TT.daily * 0.2 > 90;
      +-----------+---------+-----+
      |学号        |课程号    | 总评 |
      +-----------+---------+-----+
      | 18125111109 | c08106 | 95.0 |
      | 18125111109 | c08123 | 90.6 |
      | 18135222201 | c05109 | 93.4 |
```

```
| 18137221508  | c08106     | 92.0  |
| 18137221508  | c08171     | 96.0  |
| 19112100072  | c06108     | 97.0  |
| 19112111208  | c06108     | 93.8  |
| 19122203567  | c05103     | 91.4  |
| 19123567897  | c06127     | 99.0  |
+-----------+---------+-----+
9 rows in set (0.00 sec)
```

5.5.3 where 子句中的子查询

where 语句中的子查询实际上是将子查询的结果作为该语句条件中的一部分,然后利用这个条件过滤本层查询的数据。



where 子句中的子查询

1. 带比较运算符的子查询

子查询可以作为动态表达式,该表达式可以随着外层查询的每一行的变化而变化。即查询处理器为外部查询的每一行计算子查询的值,每次计算一行,而该子查询每次都会作为该行的一个表达式取值并返回到外层查询。使得动态执行的子查询与外部查询有一个非常有效的连接,从而将复杂的查询分解为多个简单而相互关联的查询。查询可以使用比较运算符。这些比较运算符包括=、!=、>、>=、<、<=等。比较运算符在子查询时使用得非常广泛。

创建关联子查询时,外部查询有多少行,子查询就执行多少次。

【例 5-31】 查询期末成绩比选修该课程平均期末成绩低的学生的学号、课程号和期末成绩。

分析:在本例中,对 score 表采用别名形式,一个表就相当于两个表。子查询执行时使用的 a. courseno 相当于一个常量。在别名为 b 的表中根据分组计算平均分。然后与外层查询的值进行比较,该过程很费时间。

代码和运行结果如下:

```
mysql> select studentno,courseno,final
    -> from score as a
    -> where final < (select avg(final)
    ->                from score as b
    ->                where a.courseno = b.courseno
    ->                group by courseno );
+-----------+--------+-----+
| studentno     | courseno | final |
+-----------+--------+-----+
| 18122221324  | c05103   | 62.0  |
| 18122221324  | c05109   | 77.0  |
| 18125121107  | c05109   | 62.0  |
| 18135222201  | c08171   | 82.0  |
| 18137221508  | c08106   | 95.0  |
| 18137221508  | c08123   | 89.0  |
| 19111133071  | c05103   | 69.0  |
| 19111133071  | c05109   | 82.0  |
| 19122111208  | c06127   | 67.0  |
| 19123567897  | c05103   | 77.0  |
| 19126113307  | c06108   | 82.0  |
```

```
| 19126113307  | c08171  | 79.0  |
+-------------+--------+------+
12 rows in set (0.05 sec)
```

2. 带 in 关键字的子查询

当子查询返回的结果列包含一个值时，比较运算符就符合查询要求。假如一个子查询返回的结果集是值的列表，这时比较运算符就可以用 in 运算符代替。

in 运算符可以检测结果集中是否存在某个特定的值，如果检测成功就执行外部的查询。not in 的作用与 in 刚好相反。

【例 5-32】 获取期末成绩中含有高于 93 分的学生的学号、姓名、电话和 Email。

分析：利用操作符 in 可以允许指定一个表达式(或常量)集合，可以利用 select 语句的子查询输出表达式(或常量)集合。

代码和运行结果如下：

```
mysql> select studentno,sname,phone,Email
    -> from student
    -> where studentno in ( select studentno
    ->                       from score
    ->                       where final > 93);
+-------------+--------+-------------+---------------+
| studentno   | sname  | phone       | Email         |
+-------------+--------+-------------+---------------+
| 18125111109 | 敬横江 | 15678945623 | jing@sina.com  |
| 18137221508 | 赵临江 | 12367823453 | ping@163.com   |
| 19112100072 | 宿沧海 | 12545678998 | su12@163.com   |
| 19112111208 | 韩山川 | 15878945612 | han@163.com    |
| 19122203567 | 封月明 | 13245674564 | jiao@126.com   |
| 19123567897 | 赵既白 | 13175689345 | pingan@163.com |
+-------------+--------+-------------+---------------+
6 rows in set (0.00 sec)
```

3. 带 exists 关键字的子查询

使用 exists 关键字时，内层查询语句不返回查询的记录，而是返回一个真假值。如果内层查询语句查询到满足条件的记录，就返回一个真值(true)，否则，将返回一个假值(false)。当返回的值为 true 时，外层查询语句将进行查询；当返回的值为 false 时，外层查询语句不进行查询或者查询不出任何记录。not exists 与 exists 的工作方式类似，即当 not exists 与 exists 刚好相反，使用 not exists 关键字时，当返回的值是 true 时，外层查询语句不执行查询；当返回值是 false 时，外层查询语句将执行查询。

【例 5-33】 查询 student 表中是否存在 2001 年 12 月 12 日以后出生的学生，如果存在，输出学生的学号、姓名、生日和电话。

分析：只要存在一行数据符合条件，则 where 条件就返回 true，于是输出所有行。

代码和运行结果如下：

```
mysql> select studentno,sname,birthdate,phone
    -> from student
    -> where exists (
```

```
    -> 	select *
    -> 	from student
    -> 	where birthdate < '2001-12-12');
+-------------+---------+------------+-------------+
| studentno   | sname   | birthdate  | phone       |
+-------------+---------+------------+-------------+
| 18122210009 | 许东山  | 1999-11-05 | 13623456778 |
| 18122221324 | 何白露  | 2000-12-04 | 13178978999 |
| 18125111109 | 敬横江  | 2000-03-01 | 15678945623 |
| 18125121107 | 梁一苇  | 1999-09-03 | 13145678921 |
| 18135222201 | 凌浩风  | 2001-10-06 | 15978945645 |
| 18137221508 | 赵临江  | 2000-02-13 | 12367823453 |
| 19111133071 | 崔依歌  | 2001-06-06 | 15556845645 |
| 19112100072 | 宿沧海  | 2002-02-04 | 12545678998 |
| 19112111208 | 韩山川  | 2001-02-14 | 15878945612 |
| 19122203567 | 封月明  | 2002-09-09 | 13245674564 |
| 19123567897 | 赵既白  | 2002-08-04 | 13175689345 |
| 19126113307 | 梅惟江  | 2003-09-07 | 13245678543 |
+-------------+---------+------------+-------------+
12 rows in set (0.03 sec)
```

4. 对比较运算进行限制的子查询



对比较运算进行限制的子查询

all、some 和 any 运算都是对比较运算的进一步限制。all 指定表达式要与子查询结果集中的每个值都进行比较，当表达式与每个值都满足比较的关系时，才返回 true，否则返回 false。some 或 any 是同义词，表示表达式只要与子查询结果集中的某个值满足比较的关系时，就返回 true，否则返回 false。

【例 5-34】 查找 score 表中所有比 c05109 课程期末成绩都高的学号、姓名、电话和期末成绩。

分析：本题输出项是学号、姓名、电话和期末成绩，分别存在于 student 表和 score 表，因此外层查询先做一个内连接。在此基础上，从外层查询数据源中找出每一个期末成绩 final 的值，让该值分别与子查询中的 c05109 课程的每一个值进行比较，当该外层 final 值比内层的每一个 c05109 课程成绩都高时，即为查询结果集中的一行记录。以此类推，即可得到本题的结果集。

代码和运行结果如下：

```
mysql> select student.studentno,sname, phone,final
    -> from score inner join student
    -> 	on score.studentno = student.studentno
    -> where final > all
    -> 		(select final from score where courseno = 'c05109');
+-------------+--------+-------------+-------+
| studentno   | sname  | phone       | final |
+-------------+--------+-------------+-------+
| 18125111109 | 敬横江 | 15678945623 | 99.0  |
| 18137221508 | 赵临江 | 12367823453 | 95.0  |
| 18137221508 | 赵临江 | 12367823453 | 98.0  |
| 19112100072 | 宿沧海 | 12545678998 | 97.0  |
```

```
| 19112111208  | 韩山川  | 15878945612 | 95.0   |
| 19122203567  | 封月明  | 13245674564 | 98.0   |
| 19123567897  | 赵既白  | 13175689345 | 99.0   |
+------------+-------+----------+------+
7 rows in set (0.03 sec)
```

如果将本段代码的 all 关键字换成 any 或 some 关键字，在进行数值比较时，外层查询数据源中每一个期末成绩 final 的值，只要比内层查询中的任何一个 c05109 课程成绩高，即为查询结果集中的一行记录。

5.5.4 利用子查询插入、更新与删除数据

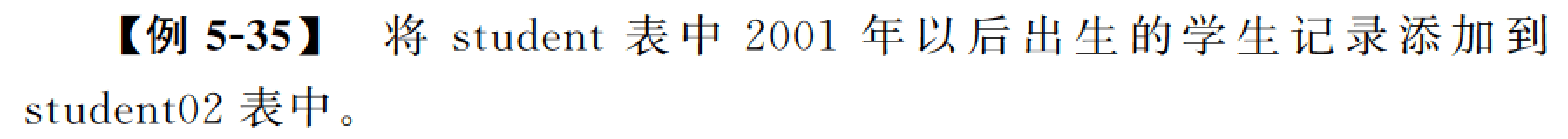
利用子查询修改表数据，就是利用一个嵌套在 insert、update 或 delete 语句的子查询成批地添加、更新和删除表中的数据。



利用子查询处理数据

1. 利用子查询插入记录

insert 语句中的 select 子查询可用于将一个或多个其他的表或视图的值添加到表中。使用 select 子查询可同时插入多行。



【例 5-35】 将 student 表中 2001 年以后出生的学生记录添加到 student02 表中。

分析：子查询的选择列表必须与 insert 语句列的列表匹配。如果 insert 语句没有指定列的列表，则选择列表必须与正向其插入的表或视图的列匹配且顺序一致。

代码和运行结果如下：

```
mysql> insert into mysqltest.student02
    ->     (select * from student
    ->     where birthdate>='2001-12-31');
    Query OK, 4 rows affected (0.49 sec)
    Records: 4 Duplicates: 0 Warnings: 0
```

2. 利用子查询更新数据

update 语句中的 select 子查询可用于将一个或多个其他的表或视图的值进行更新。使用 select 子查询可同时更新多行数据。实际上是通过将子查询的结果作为更新条件表达式中的一部分。

【例 5-36】 将 student 表中入学成绩低于 800 分的所有学生的期末成绩增加 5%。

分析：利用 update 成批修改表数据，可以在 where 子句中利用子查询实现。

代码和运行结果如下：

```
mysql> update score
    -> set final= final*1.05
    -> where studentno in
    ->     (select studentno
    ->     from student
    ->     where entrance<800);
    Query OK, 18 rows affected (0.03 sec)
    Rows matched: 18 Changed: 18 Warnings: 0
```

同样在 delete 语句中利用子查询可以删除符合条件的数据行。实际上是通过将子查询的结果作为删除条件表达式中的一部分。

5.6 使用正则表达式进行模糊查询


正则表达式

正则表达式通常用来检索或替换符合某个模式的文本内容，根据指定的匹配模式匹配文本中符合要求的特殊字符串。例如从一个文本文件中提取电话号码，查找一篇文章中重复的单词或者替换用户输入的某些词语等。正则表达式强大而且灵活，可以应用于非常复杂的查询。本节将详细讲解如何使用正则表达式来查询。

正则表达式的查询能力比通配字符的查询能力更强大，而且更加灵活。正则表达式可以应用于非常复杂的查询。MySQL 中，使用 regexp 关键字来匹配查询正则表达式。正则表达式的基本语法格式如下：

```
where 字段名 regexp '操作符'
```

MySQL 中使用 regexp 操作符指定正则表达式的字符匹配模式，regexp 操作符中常用的字符匹配选项如表 5-1 所示。

表 5-1 正则表达式中常用的字符匹配选项

选　项	说　明	示　例
^	匹配文本的开始字符	^b：匹配以字母 b 为开头的字符串，如 big
$	匹配文本的结束字符	st $：匹配以 st 结尾的字符串，如 test
.	匹配任何单个字符	b. t：匹配任何 b 和 t 之间有一个字符，如 bit
*	匹配零个或多个在它前面的字符	* n：匹配字符 n 前面有任意个字符，如 fn
+	匹配前面的字符 1 次或多次	ba+：匹配以 b 开头后面紧跟至少有一个 a，如 bay、bare、battle
<字符串>	匹配包含指定的字符串的文本	fa：字符串至少要包含 fa，如 fan
[字符集合]	匹配字符集合中的任何一个字符	[xz]：匹配 x 或 z，如 dizzy
[^]	匹配不在括号中的任何字符	[^abc]：匹配任何不包含 a、b 或 c 的字符串
字符串{n,}	匹配前面的字符串至少 n 次	b{2,}：匹配两个或更多的 b，如 bb、bbb
字符串{m,n}	匹配前面的字符串至少 m 次，至多 n 次。如果 n 为 0，m 为可选参数	b{2,4}：匹配至少 2 个 b，最多 4 个 b，如 bb、bbbb、bbb

1. 查询以特定字符或字符串开头的记录

使用字符“^”可以匹配以特定字符或字符串开头的记录。

【例 5-37】 查询 student 表中姓“赵”的学生的部分信息。

代码和运行结果如下：

```
mysql > select studentno, sname, birthdate, phone
    -> from student
    -> where sname regexp '^赵';
      +------------+-------+------------+-------------+
      | studentno  | sname | birthdate  | phone       |
      +------------+-------+------------+-------------+
      | 18137221508 | 赵临江 | 2000 - 02 - 13 | 12367823453 |
      | 19123567897 | 赵既白 | 2002 - 08 - 04 | 13175689345 |
```

```
+------------+------+-----------+------------+
2 rows in set (0.00 sec)
```

2. 查询以特定字符或字符串结尾的记录

使用字符“$”可以匹配以特定字符或字符串结尾的记录。

【例 5-38】 查询 student 表中学生电话号码尾数为 5 的学生部分信息。

代码和运行结果如下：

```
mysql> select studentno, sname, phone, Email
    -> from student
    -> where phone regexp '5$';
+------------+--------+------------+-------------+
| studentno  | sname  | phone      | Email       |
+------------+--------+------------+-------------+
| 18135222201 | 凌浩风 | 15978945645 | tang@163.com |
| 19111133071 | 崔依歌 | 15556845645 | cui@126.com  |
| 19123567897 | 赵既白 | 13175689345 | pingan@163.com |
+------------+--------+------------+-------------+
3 rows in set (0.00 sec)
```

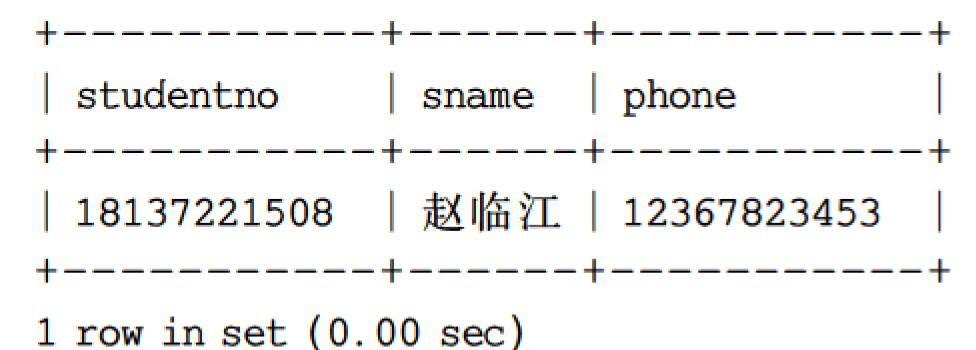
3. 用符号“.”来替代字符串中的任意一个字符

用正则表达式来查询时，可以用“.”来替代字符串中的任意一个字符。

【例 5-39】 要实现查询学生姓名 sname 字段中以“赵”开头，以“江”结束，中间包含两个字符的学生信息，可以通过正则表达式查询来实现，其中正则表达式中“^”表示字符串的开始位置，$ 表示字符串的结束位置，. 表示除“\n”以外的任何单个字符（此例中汉字按两个字符计算）。

代码和运行结果如下：

```
mysql> select studentno, sname, phone
    -> from student
    -> where sname regexp '^赵..江$';
```



studentno	sname	phone
18137221508	赵临江	12367823453

1 row in set (0.00 sec)

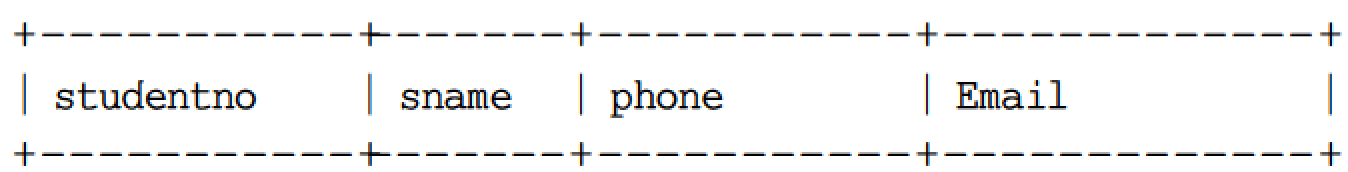
4. 匹配指定字符串

正则表达式可以匹配字符串。当表中的记录包含这个字符串时，就可以将该记录查询出来。如果指定多个字符串，需要用符号“|”隔开。只要匹配这些字符串中的任意一个即可。

【例 5-40】 查询学生电话号码出现 131 或 132 数字的学生信息。

代码和运行结果如下：

```
mysql> select studentno, sname, phone, Email
    -> from student
    -> where phone regexp '131|132';
```



studentno	sname	phone	Email

```
| 18122221324  | 何白露 | 13178978999 | heyy@sina.com   |
| 18125121107  | 梁一苇 | 13145678921 | bing@126.com    |
| 19122203567  | 封月明 | 13245674564 | jiao@126.com    |
| 19123567897  | 赵既白 | 13175689345 | pingan@163.com  |
| 19126113307  | 梅惟江 | 13245678543 | zhu@163.com     |
+-------------+-------+-------------+-----------------+
5 rows in set (0.00 sec)
```

由于 MySQL 中不同字符集的影响，在使用正则表达式时需要多练习，多上机实验才能更好地掌握，并实现举一反三的学习效果。

5.7 小　　结

本章介绍了 MySQL 数据库常见的查询方法。利用选择、投影和连接理论知识，实现对指定字段、指定记录和使用 like 关键字及通配符的查询，以及使用 and 和 or 来实现多条件查询、分组查询、连接查询、子查询等。分组查询经常和聚合函数一起使用，而且使用方法非常灵活。使用 limit 关键字来限制查询结果的条数是 MySQL 数据库的特色。本章的难点是使用正则表达式来查询。正则表达式的功能很强大，使用起来很灵活。具体需要掌握的主要内容如下：

- select 语句的一般格式及各个子句的作用。
- 在 where 子句中使用 like、in、between 关键字时的各种操作。
- select 语句中利用聚合函数实现计算和统计操作。
- 连接查询的格式、分类和应用。
- select 语句中使用子查询的技巧。
- 学习使用正则表达式进行查询。

习　题　5

1. 选择题

(1) select 语句中使用________关键字可以将重复行屏蔽。

A. order by　　B. having　　C. top　　D. distinct

(2) 在 select 语句中，可以使用________子句，将结果集中的数据行根据选择列的值进行逻辑分组，以便能汇总表内容的子集，即实现对每个组的聚集计算。

A. limit　　B. group by　　C. where　　D. order by

(3) 使用空值查询时，表示一个列 RR 不是空值的表达式是________。

A. RR is null　　B. RR = null　　C. RR <> null　　D. RR is not null

(4) select　*　from　city　limit 5,10 描述正确的是________。

A. 获取第 6 条到第 10 条记录　　B. 获取第 5 条到第 10 条记录

C. 获取第 6 条到第 15 条记录　　D. 获取第 5 条到第 15 条记录

(5) select 语句中用于实现关系的选择运算的短语是________。

A. for　　B. while　　C. where　　D. condition

(6) 关于 select 语句以下哪一个描述是错误的? ________

A. select 语句用于查询一个表或多个表的数据

B. select 语句属于数据操作语言(DML)

C. select 语句的输出列必须是基于表的列

D. select 语句表示数据库中一组特定的数据记录

(7) select 语句的执行过程是从数据库中选取匹配的特定记录和字段,并将这些数据组织成一个结果集,然后以________的形式返回。

A. 结构体数组　　B. 系统表　　C. 永久表　　D. 临时表

(8) 现有订单表 orders,包含用户信息 userid,产品信息 productid,以下________语句能够返回至少被订购过两回的 productid。

A. select productid from orders where count(productid)>1

B. select productid from orders where max(productid)>1

C. select productid from orders where having count(productid)>1 group by productid

D. select productid from orders group by productid having count(productid)>1

2. 思考题

(1) 简述 select 语句的各个子句的作用。

(2) MySQL 中通配符与正则表达式的区别是什么?

(3) 说明在 select 语句中使用聚合函数应该注意的问题。

(4) 将 null 与其他值比较会产生什么结果? 数值列中存在 null 会产生什么结果?

(5) 简述连接查询和利用 union 语句合并结果集的应用区别。

(6) 什么情况下使用 limit 来限制查询结果的数量?

3. 上机练习题(本题利用 teaching 数据库进行操作)

(1) 查询 course 表中的所有记录。

(2) 查询 student 表中女生的人数。

(3) 查询 teacher 表中每一位教授的教师号、姓名和专业名称。

(4) 按性别分组,求出 student 表中每组学生的平均年龄。

(5) 创建新表,新表中包括学号、学生姓名、课程号和总评成绩。其中,总评成绩=final * 0.8+daily * 0.2 。

(6) 统计每个学生的期末成绩平均分。

(7) 输出 student 表中年龄最大的男生的所有信息。

(8) 查询 teacher 表中没有职称的职工的教师号、姓名、专业和部门。

第 6 章 索引和视图

在 MySQL 数据库中，索引(Index)是影响数据性能的重要因素之一，设计高效的、合理的索引可以显著提高数据信息的查询速度和应用程序的性能。

视图(View)是一个存储指定查询语句的虚拟表，视图中数据来源于由定义视图所引用的表，并且能够实现动态引用，即表中数据发生变化，视图中的数据随之变化。

本章将介绍索引和视图等数据库对象的基本概念和常用操作。

6.1 索　　引

索引由数据库表中一列或多列组合而成的一种特殊的数据库结构，利用索引可以快速查询数据库表中的特定记录信息。在 MySQL 中，所有的数据类型都可以被索引。

6.1.1 理解索引

MySQL 的索引是为了加速对数据进行检索而创建的一种分散的、物理的数据结构。索引包含从表或视图中一个或多个列生成的键，以及映射到指定数据行的存储位置指针。索引是依赖于表建立的，提供了数据库中编排表中数据的内部方法。表的存储由两部分组成，一部分是表的数据页面，另一部分是索引页面，索引就存放在索引页面上。

数据库中的索引的形式与图书的目录相似，键值就像目录中的标题，指针相当于页码。索引的功能就像图书目录能为读者提供快速查找图书页面内容一样，不必扫描整个数据表而找到想要的数据行。

可以想象一下，当 MySQL 数据库在执行一条查询语句的时候，默认的执行过程是根据搜索条件进行全表扫描，遇到匹配条件的就加入搜索结果集合。如果查询语句涉及多个表连接，包括了许多搜索条件(例如大小比较、like 匹配等)，而且表数据量特别大时，在没有索引的情况下，MySQL 需要执行的扫描行数会很多，速度也会很慢。

索引一旦创建，将由数据库自动管理和维护。例如，向表中插入、更新和删除一条记录时，数据库会自动在索引中做出相应的修改。在编写 SQL 查询语句时，具有索引的表与不具有索引的表没有任何区别，索引只是提供一种快速访问指定记录的方法。

实际过程中，当 MySQL 执行查询时，查询优化器会对可用的多种数据检索方法的成本进行估计，从中选用最有效的查询计划。

在数据库中使用索引的优点如下。

(1) 加速数据检索：索引能够以一列或多列值为基础实现快速查找数据行。

(2) 优化查询：查询优化器是依赖于索引起作用的，索引能够加速连接、排序和分组等

操作。

(3) 强制实施行的唯一性：通过给列创建唯一索引，可以保证表中的数据不重复。

需要注意的是，索引并不是越多越好，要正确认识索引的重要性和设计原则，创建合适的索引。

6.1.2 索引的分类

按照分类标准的不同，MySQL 的索引有多种分类形式。

MySQL 的索引通常包括普通索引、唯一性索引、主键索引、全文索引和空间索引等类型。

(1) 普通索引(index)。索引的关键字是 index。普通索引是 MySQL 中的基本索引类型，允许在定义索引的列中插入重复值和空值。

(2) 主键索引(primary key)。主键索引是一种特殊的唯一索引，不允许有空值。一般是在建表的时候同时创建主键索引。也可通过修改表的方法增加主键，但一个表只能有一个主键索引。

(3) 唯一性索引(unique)。unique 索引列的值必须唯一，允许有空值。如果是组合索引，则列值的组合必须唯一，在一个表上可以创建多个唯一性索引。

(4) 全文索引(fulltext)。全文索引是指在定义索引的列上支持值的全文查找，允许在这些索引列中插入重复值和空值。该索引只能对 char、varchar 和 text 类型的列编制索引，并且只能在 MyISAM 表中编制。即 MySQL 中只有 MyISAM 存储引擎支持全文索引。在 MySQL 默认情况下，对于中文作用不大。

(5) 空间索引(spatial)。空间索引是对空间数据类型的字段建立的索引。MySQL 中的空间数据类型有 4 种，分别是 geometry、point、linestring 和 polygon。MySQL 使用 spatil 关键字进行扩展，使得能够用于创建正规索引类似的语法创建空间索引。创建空间索引的列，必须将其声明为 not null，空间索引只有在存储引擎 MyISAM 的表中创建。对于初学者来说，这类索引很少会用到。

如果按照创建索引键值的列数分类，索引还可以分为单列索引和复合索引。

如果按照存储方式分类，MySQL 的索引分为 B-Tree 索引和 Hash 索引。

目前许多主流数据库管理系统如 Oracle、SQL Server、MySQL 等，都是将 B-Tree 索引作为最主要的索引类型，这主要是因为 B-Tree 索引的存储结构在数据库的数据检索中有着非常优异的表现。

MySQL 的 MEMORY 数据引擎还支持 Hash 索引。Hash 索引相对于 B-Tree 索引，检索效率要高很多。但 MySQL Hash 索引本身的特殊性也带来了很多限制和弊端，主要有以下内容。

- 仅仅能满足“=”、“in”和“<=>”查询，不能使用范围查询。
- 无法被用来避免数据的排序操作。
- 不能利用部分索引键查询。
- 在任何时候都不能避免表扫描。
- 遇到大量 Hash 值相等的情况后，性能并不一定就会比 B-Tree 索引高。

说明：

MySQL 中运算符“ ＜＝＞”除了能够像常规的“＝”运算符一样，对两个值进行比较外，还能够用于比较 null 值。

- 运算符“ ＜＝＞”和“＝”的相同点。像常规的“＝”运算符一样，两个值进行比较，结果是 0（不相等）或 1（相等）。例如，'A'＜＝＞'B'的值为 0，'a'＜＝＞'a'的值为 1。
- 运算符“ ＜＝＞”和“＝”号的不同点。和“＝”运算符不同的是，null 与常量进行比较运算，其值直接处理为 null。使用“ ＜＝＞”运算符时，例如，'a' ＜＝＞ null 的值为 0，null＜＝＞ null 的值为 1；相当于'a' is null 和 null is null。而'a' is not null 则相当于 not('a' ＜＝＞ null)。

6.1.3 设置索引的原则

在数据表中创建索引，为使索引的使用效率更高，必须考虑在哪些字段上创建索引和创建什么类型的索引。首先要了解以下常用的基本原则。

(1) 一个表创建大量索引，会影响 insert、update 和 delete 语句的性能。应避免对经常更新的表创建过多的索引，要限制索引的数目。

(2) 若表的数据量大，对表数据的更新较少而查询较多，可以创建多个索引来提高性能。在包含大量重复值的列上创建索引，查询的时间会较长。

(3) 经常需要排序、分组和联合操作的字段一定要建立索引，即将用于 join、where 判断和 order by 排序的字段上创建索引。

(4) 在视图上创建索引可以显著地提升查询性能。

(5) 尽量不要对数据库中某个含有大量重复值的字段建立索引，在这样的字段上建立索引有可能降低数据库的性能。

(6) 在主键上创建索引，在 InnoDB 中如果通过主键来访问数据效率是非常高的。每个表只能创建一个主键索引。

(7) 要限制索引的数目，对于不再使用或者很少使用的索引要及时删除。

(8) InnoDB 数据引擎的索引键最长支持 767 字节，MYISAM 数据引擎支持 1000 字节。

6.1.4 创建索引

创建索引通常有三种命令方式，即创建表时附带创建索引、通过修改表来创建索引和使用 alter table 语句来创建索引。当然利用 MySQL Workbench 等工具也可以实现可视化方式创建索引，下面将详细讲解这些创建索引的方法。



创建索引

1. 利用 create index 语句创建三种索引

如果基表已经创建完毕，就可以使用 create index 语句创建索引。

创建索引基本形式如下：

```
create [unique|fulltext|spatial] index index_name
on table_name (index_col_name,...)
```

说明：

(1) create index ：创建索引的关键词。

(2) unique|fulltext|spatial：创建索引的类型。unique 是唯一索引，fulltext 是全文索引，spatial 是空间索引。

(3) index_name：索引名。索引名可以不写，若不写索引名，则默认与列名相同。

(4) on table_name ：创建索引对应的表。

(5) index_col_name：index_col_name 的格式如下：

```
col_name [(length)] [asc | desc]
```

创建索引时，可以使用 col_name(length)语法对前缀编制索引。前缀包括每列值的前 length 个字符。对于 char 和 varchar 列，只用一列的一部分就可创建索引。blob 和 text 列也可以编制索引，但是必须给出前缀长度。因为多数名称的前 10 个字符通常不同，所以创建索引不会比使用列的全名创建索引的速度慢很多。

另外，使用列的一部分创建索引可以使索引文件大大减小，从而节省大量的磁盘空间，有可能提高 insert 操作的速度。

【例 6-1】 为便于按电话进行查询，在 student 表的 phone 列上建立一个升序普通索引 phone_index。

代码和运行结果如下：

```
mysql> use teaching;
        Database changed
mysql> create index phone_index on student(phone asc);
        Query OK, 0 rows affected (0.80 sec)
        Records: 0 Duplicates: 0 Warnings: 0
```

【例 6-2】 在 course 表的 cname 列上建立一个唯一性索引 cname_index。

代码和运行结果如下：

```
mysql> create unique index cname_index on course (cname);
        Query OK, 0 rows affected (0.44 sec)
        Records: 0 Duplicates: 0 Warnings: 0
```

【例 6-3】 在 score 表的 studentno 和 courseno 列上建立一个复合索引 sc_index。

代码和运行结果如下：

```
mysql> create index sc_index on score(studentno,courseno);
        Query OK, 0 rows affected (0.23 sec)
        Records: 0 Duplicates: 0 Warnings: 0
```

2. 创建表时创建索引

创建表时可以直接创建索引，这种方式最简单、方便。

【例 6-4】 创建 teacher1 表的 tname 字段建立一个唯一性索引 tname_index，一个前缀索引 dep_index。

代码和运行结果如下：

```
mysql> use mysqltest;
        Database changed
mysql> create table if not exists teacher1 (
```

```
    -> teacherno char(6) not null comment '教师编号',
    -> tname char(8) not null comment'教师姓名',
    -> major char(10) not null comment '专业',
    -> prof char(10) not null comment '职称',
    -> department char(16) not null comment '部门',
    -> primary key (teacherno),
    -> unique index tname_index(tname),
    -> index dep_index(department(5))
    -> );
       Query OK, 0 rows affected (0.41 sec)
```

3. 通过 alter table 语句创建索引

【例 6-5】 在 teacher1 表上建立 teacherno 主键索引(假定未创建主键索引),建立 tname 和 prof 的复合索引。

代码和运行结果如下:

```
mysql> alter table teacher1
    -> add primary key(teacherno),
    -> add index mark(tname, prof);
       Query OK, 0 rows affected (0.71 sec)
       Records: 0 Duplicates: 0 Warnings: 0
```

如果主键索引已经创建,则会出现如下信息:

```
ERROR 1068 (42000): Multiple primary key defined
```

说明:

(1) 只有表的所有者才能给表创建索引。索引的名称必须符合 MySQL 的命名规则,且必须是表中唯一的。

(2) 主键索引必定是唯一的,唯一性索引不一定是主键。一张表上只能有一个主键,但可以有一个或者多个唯一性索引。

(3) 当给表创建 unique 约束时,MySQL 会自动创建唯一性索引。创建唯一性索引时,应保证创建索引的列不包括重复的数据,并且没有两个或两个以上的空值(null)。因为创建索引时将两个空值也视为重复的数据,如果有这种数据,必须先将其删除,否则索引不能被成功创建。

(4) 若要查看表中已经创建索引的情况,可以使用 show index from table_name 语句实现。

6.1.5 删除索引

删除不再需要的索引,可以通过 drop 语句来删除索引,也可用 alter table 语句删除。利用 drop index 语句删除索引的语法格式如下:

```
drop index index_name on table_name ;
```

例如,删除 teacher1 表的 mark 索引:

```
mysql> drop index mark on teacher1;
```

利用 alter table 语句删除索引的语法格式如下:

```
alter [ignore] table table_name
| drop primary key
| drop index index_name
| drop foreign key fk_symbol
```

利用 alter table 语句同样可以删除前面表中创建的索引。例如：

```
mysql> alter table course drop index cname_index;
```

说明：

(1) drop index 子句可以删除各种类型的索引，包括唯一索引。

(2) 如要删除主键索引，则直接使用 drop primary key 子句进行删除，不需要提供索引名称，因为一个表中只有一个主键。

6.1.6 利用 MySQL Workbench 工具创建和管理索引



利用 Workbench 管理索引

1. 利用 MySQL Workbench 创建索引

(1) 启动 MySQL Workbench 工具，单击实例 mysql57。在导航区 Navigator 下的 SCHEMAS 区域，选择当前数据库 teaching。

(2) 在 teaching 数据库中选择 tables，展开 tables 选项，选择表 student，如图 6-1 所示，在弹出的菜单中执行 Alter Table 命令，如图 6-2 所示。

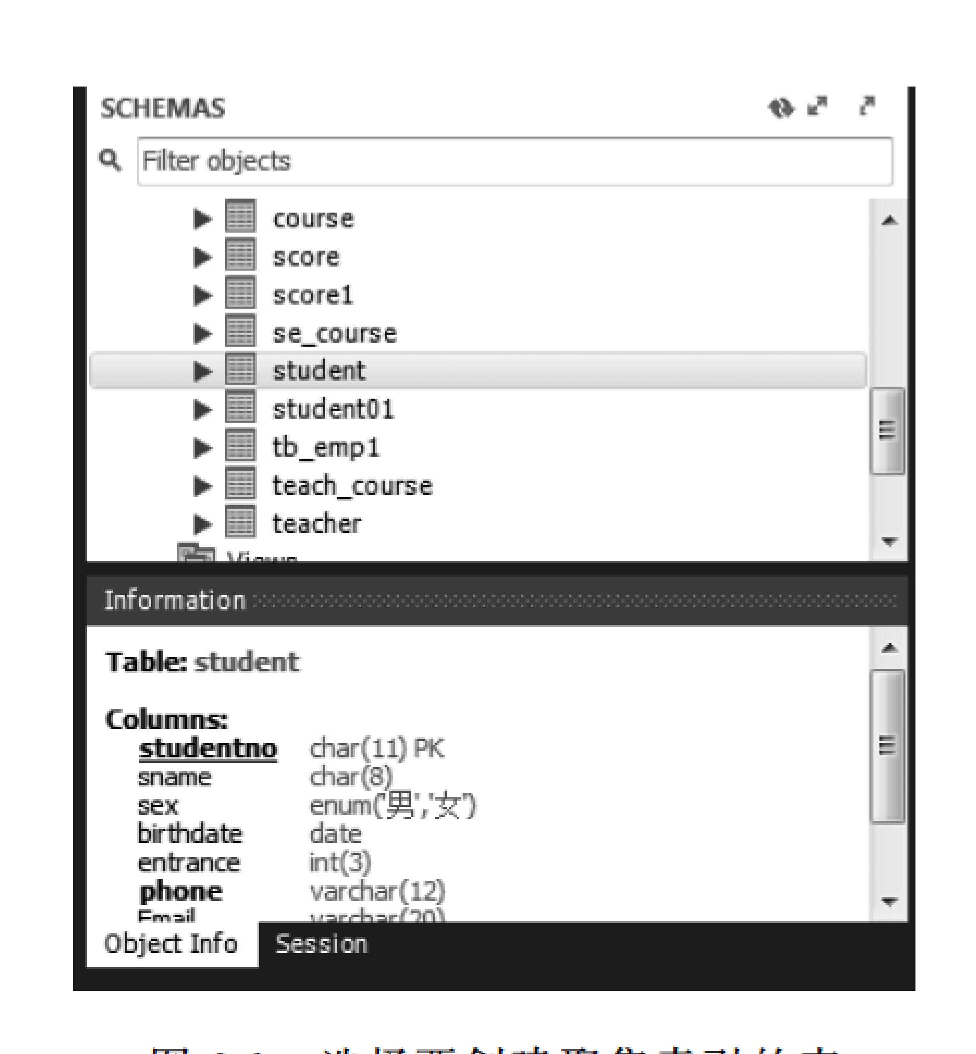


图 6-1 选择要创建聚集索引的表

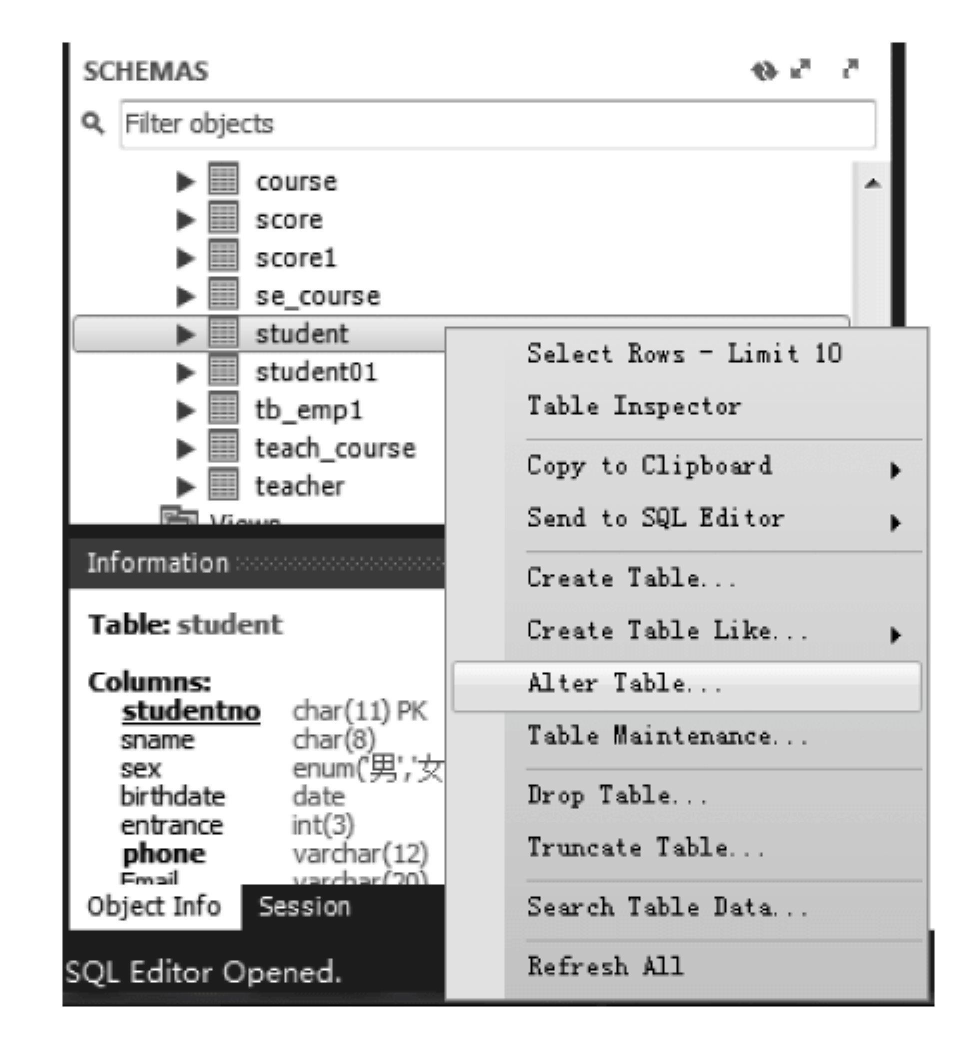


图 6-2 执行修改表命令

(3) 进入修改表 student 界面，如图 6-3 所示。打开 indexes 选项卡，在如图 6-4 所示的界面中可以观察到如下信息。

① 数据库名和表名：指出创建索引的数据库 teaching 和表 student 的名称。

② 表 student 的默认字符集及排序规则和数据引擎 InnoDB 索引名称。

③ 索引名 Index Name，可以查看到前面创建的主键索引 primary 和唯一索引 phone_index。其后依次是索引类型 Type、索引引用字段 Index Columns、索引参数 Index Options 和索引注解 Index Comment 等。



图 6-3　创建索引表结构

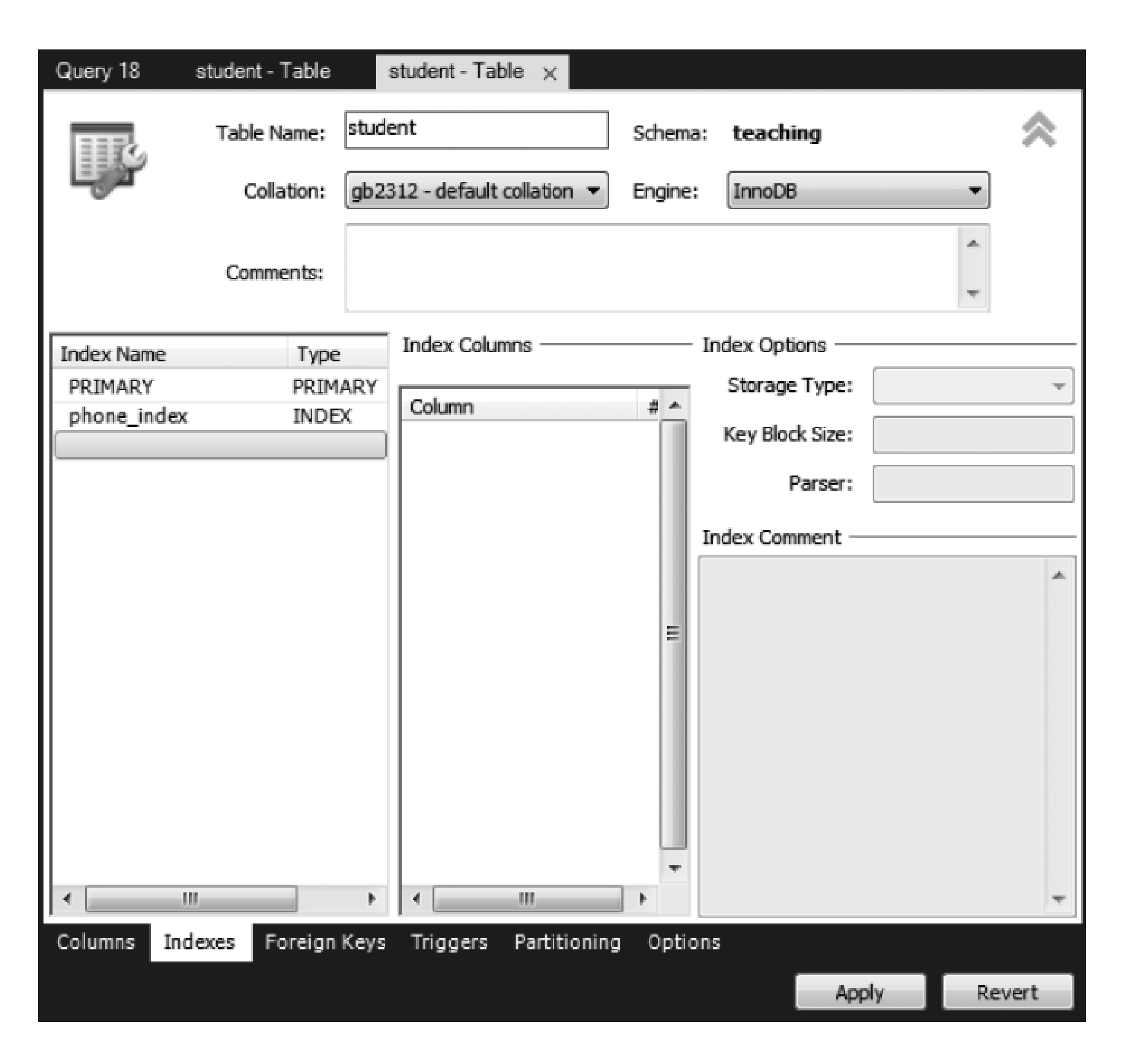


图 6-4　创建索引对话框

(4) 在 Index Name 的文本框中输入索引名称 un_phone，右侧的 Index Columns 会自动显示表 student 中的所有列名，选择 phone 列。存储类型选择 BTREE，选择索引类型 unique，表示创建唯一性索引，其他参数采用默认值，如图 6-5 所示。

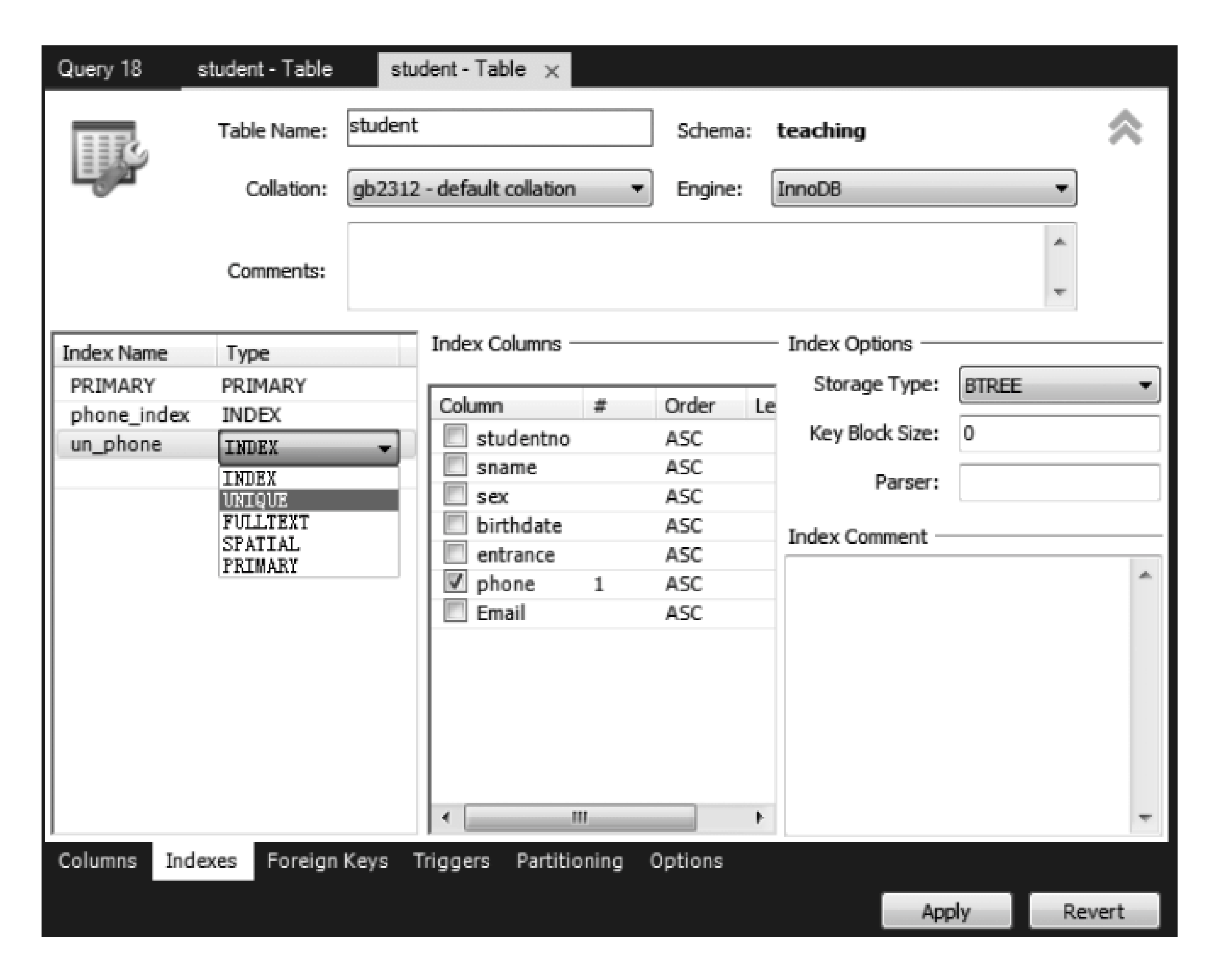


图 6-5　设置索引参数

(5) 设置完成后，单击 Apply 按钮，出现如图 6-6 所示的应用脚本对话框。再单击 Apply 按钮，进入完成对话框，单击 Finish 按钮，即可完成在数据库 teaching 中 student 表上的唯一性索引 un_phone 的创建。

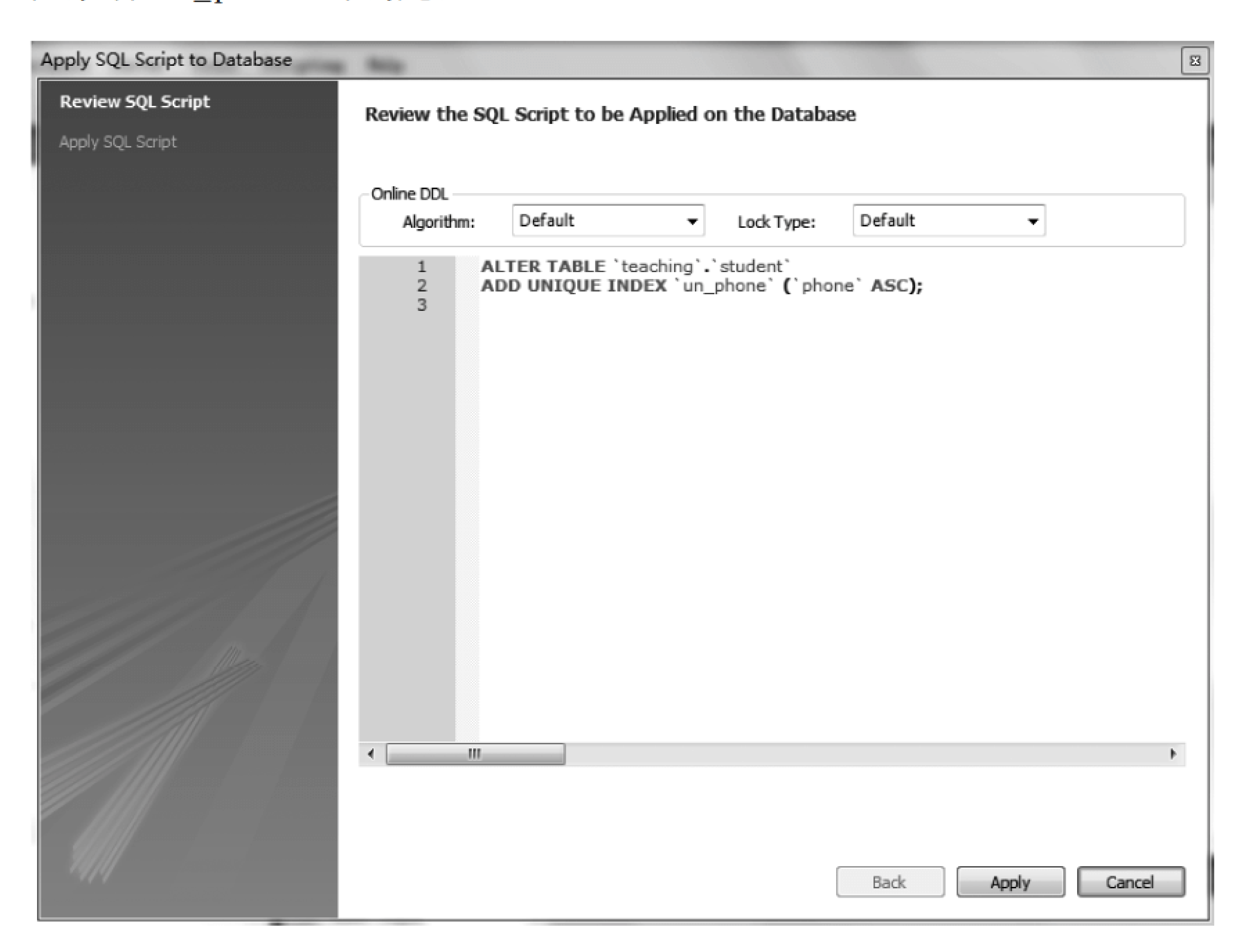


图 6-6　创建索引脚本

在 MySQL Workbench 中创建主键索引和普通索引的操作步骤基本相同。

2. 利用 MySQL Workbench 管理索引

(1) 利用 MySQL Workbench 修改索引，可以修改索引的名字、类型、索引引用字段和

索引参数等。

例如，修改 student 表中的 un_phone 索引为普通索引 un_phone_Email，索引类型改为 index，引用字段为 phone 和 Email，且为降序排列，如图 6-7 所示。单击 Apply 按钮，出现如图 6-8 所示的应用脚本对话框。再单击 Apply 按钮，进入完成对话框，如图 6-9 所示，单击 Show Logs(Hide Logs)，可以查看(隐藏)日志消息。单击 Finish 按钮，即可完成在数据库 teaching 中 student 表上的索引 un_phone 的修改。

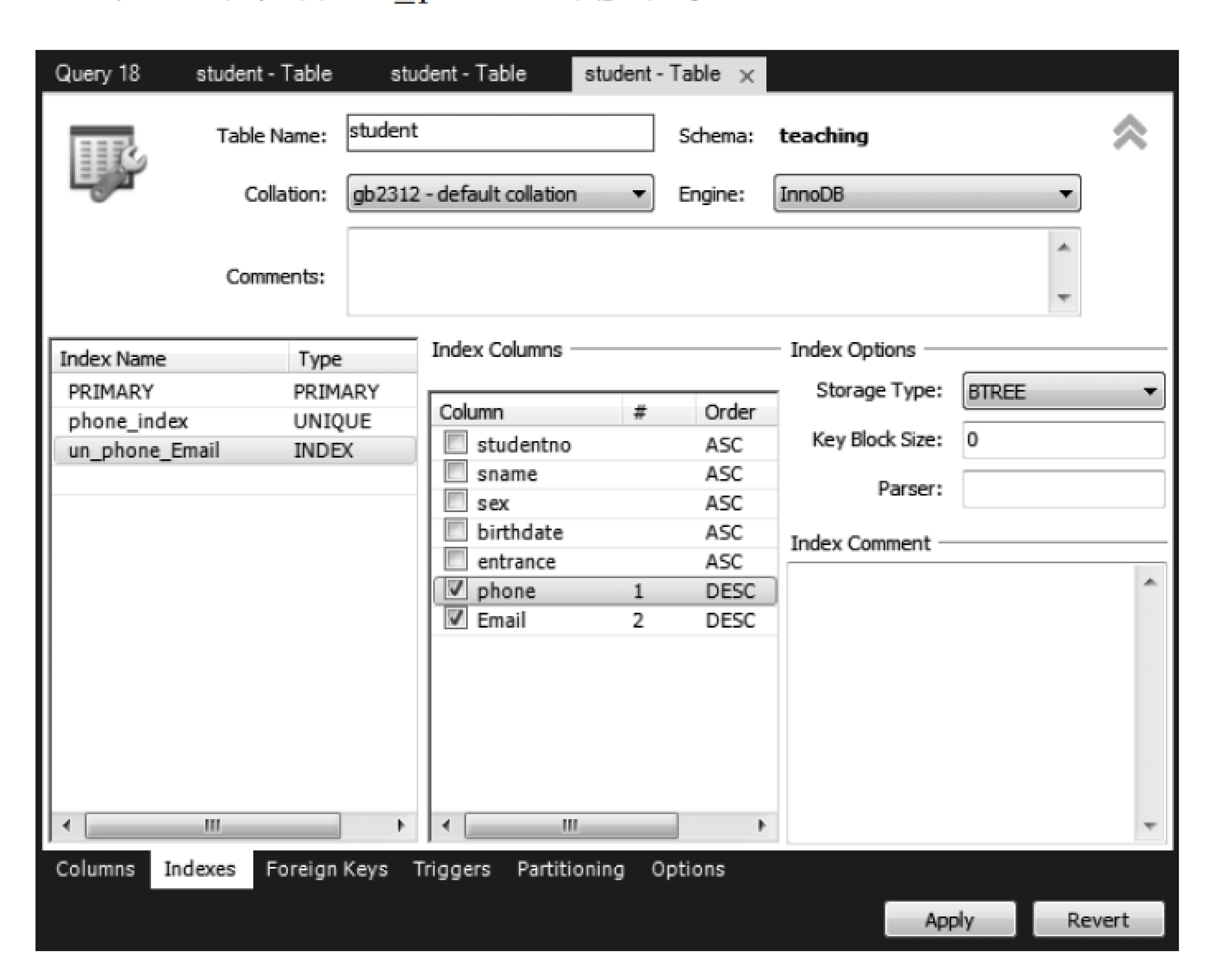


图 6-7　修改索引参数

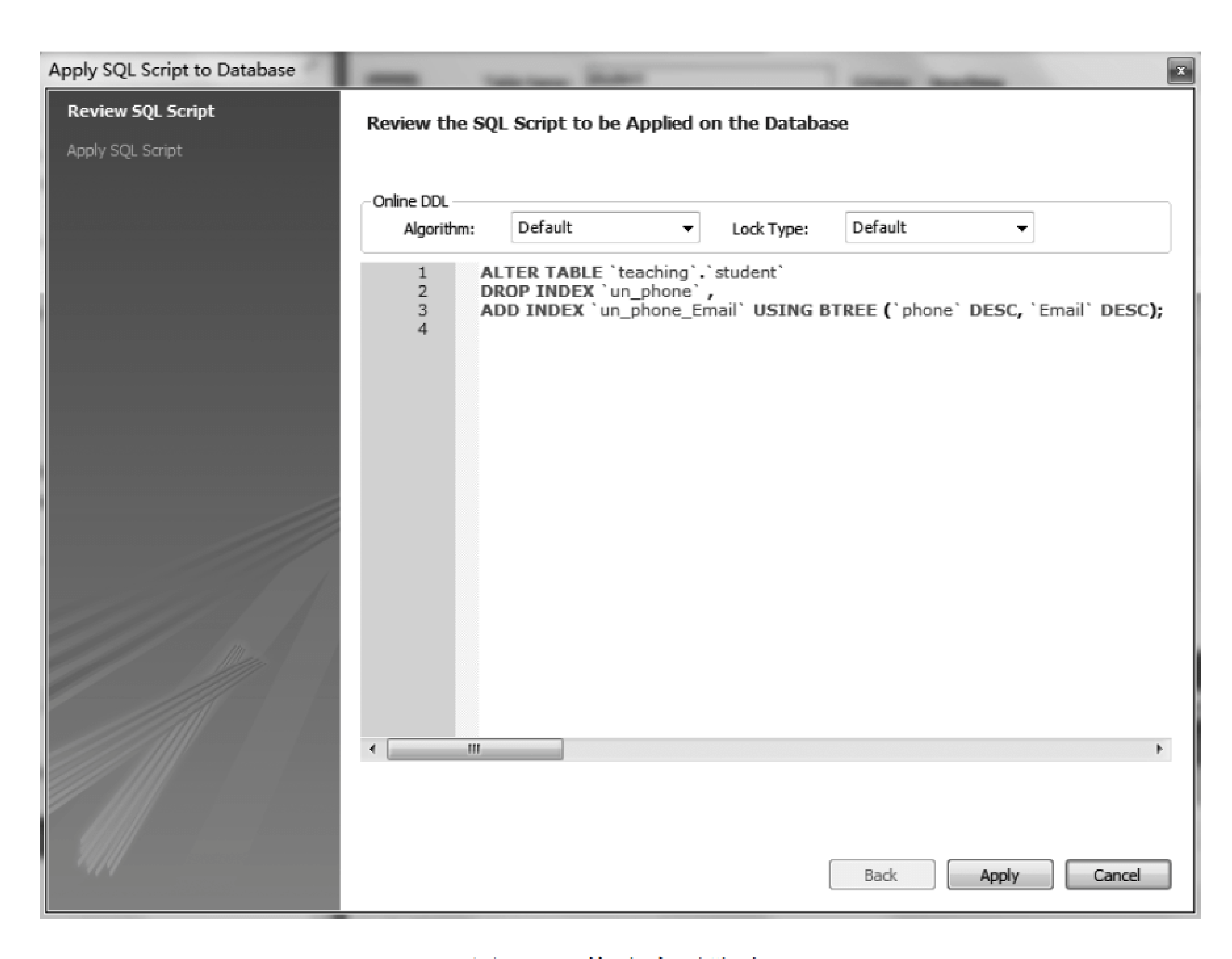


图 6-8　修改索引脚本

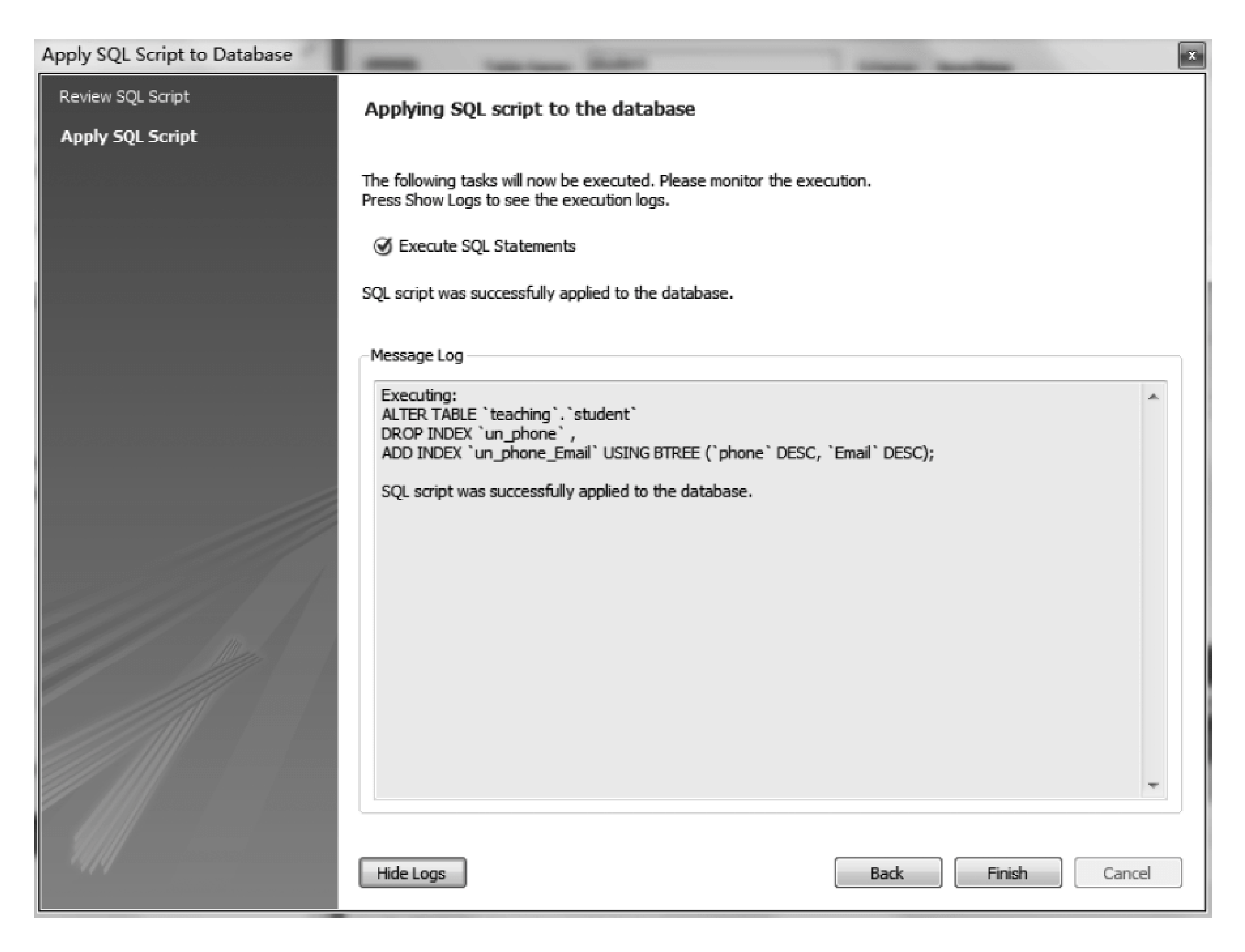


图 6-9　完成修改索引信息

(2) 利用 MySQL Workbench 删除索引。例如，删除普通索引 un_phone_Email。在索引界面中，如图 6-10 所示，右击索引 un_phone_Email，执行 Delete Selected 命令，索引 un_phone_Email 即从列表中消失。单击 Apply 按钮，出现删除索引的应用脚本对话框；再单击 Apply 按钮，进入完成对话框；单击 Finish 按钮，即可删除索引 un_phone_Email。

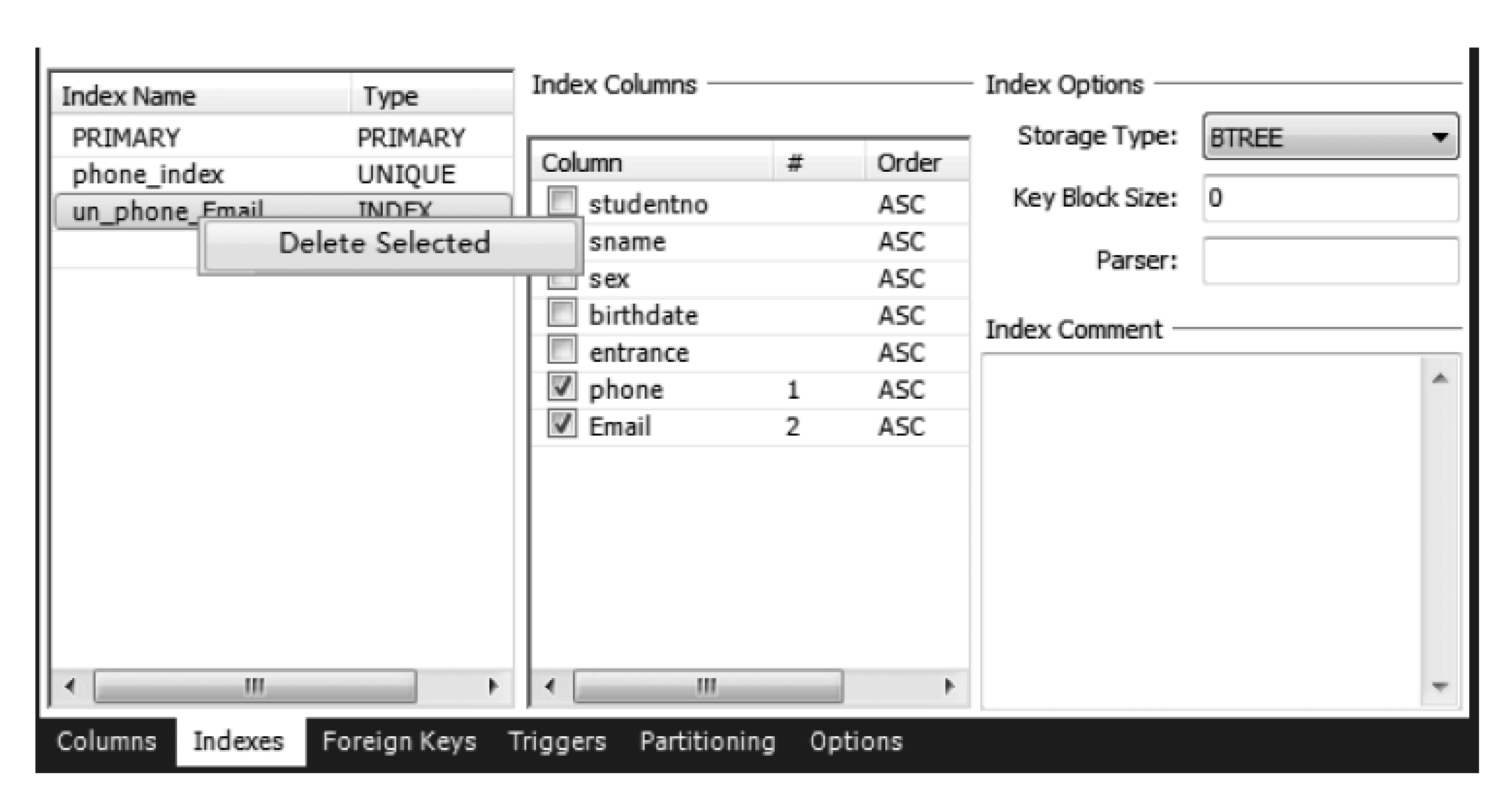


图 6-10　删除索引

6.2　视图的创建和管理

视图是从一个或者多个表及其他视图中通过 select 语句导出的虚拟表，数据库中只存放了视图的定义，而并没有存放视图中的数据。浏览视图时所对应数据的行和列数据来自

定义视图查询所引用的表，并且在引用视图时动态生成。通过视图可以实现对基表数据的查询与修改。

视图为数据库用户提供了很多的便利，主要包括以下几个方面。

(1) 简化数据查询和处理。视图可以为用户集中多个表中的数据，简化用户对数据的查询和处理。

(2) 屏蔽数据库的复杂性。数据库表的更改不影响用户对数据库的使用，用户也不必了解复杂的数据库中的表结构。例如，那些定义了若干张表连接的视图，就将表与表之间的连接操作对用户隐藏起来了。

(3) 安全性。如果想要使用户只能查询或修改用户有权限访问的数据，也可以只授予用户访问视图的权限，而不授予访问表的权限，这样就提高了数据库的安全性。

6.2.1 创建视图

创建视图是指在指定的数据库表上建立视图。视图可以创建在一张表上，也可以创建在多张表或既有视图上。要求创建用户具有针对视图的create view 权限，以及针对由 select 语句选择的每一列上的某些权限。

创建视图

1. 创建视图的语法形式

创建视图是通过 create view 实现的。其语法形式如下：

```
create [or replace][algorithm = { undefined|merge|temptable }]
view view_name [(column_list)]
as select_statement
[ with[ cascaded|local] check option ] ;
```

说明：

(1) create view 语句能创建新的视图，如果给定了 or replace 子句，该语句还能替换已有的视图。

(2) view_name 为视图名。视图属于数据库。在默认情况下，将在当前数据库中创建视图。如果要在其他给定数据库中创建视图，应将名称指定为 db_name. view_name，视图名不能与表同名。

(3) algorithm = { undefined | merge | temptable }：algorithm 为视图算法选择，有三个选项：undefined 表示 MySQL 将自动选择算法；merge 表示将合并视图定义和视图语句，使得视图定义的某一部分取代语句的对应部分；temptable 表示将视图结果存储到临时表，然后利用临时表执行语句。

(4) select_statement：用来创建视图的 select 语句，它给出了视图的定义。该语句可从基本表或其他视图进行选择。默认情况下，由 select 语句检索的列名将用作视图列名。如果想为视图列定义另外的名称，可使用可选的 column_list 子句，列出由逗号隔开的列名称即可。但要注意，column_list 中的名称数目必须等于 select 语句检索的列数。

(5) cascaded | local：为可选参数。cascaded 为默认值，表示更新视图时要满足所有相关视图和表的条件；local 表示更新视图时满足该视图本身的定义即可。

(6) [with check option] ：要求具有针对视图的 create view 语句权限，以及针对由select 语句选择列上的某些权限。对在 select 语句中使用其他来源的列，必须具有 select 语

句权限，如果还有[or replace]语句，则必须具有 drop 权限。

(7) 在视图定义中命名的表必须已存在，视图必须具有唯一的列名，不得有重复，就像基本表那样。还要有如下限制：

- 在视图的 from 子句中不能使用子查询。
- 在视图的 select 语句中不能引用系统或用户变量。
- 在视图的 select 语句中不能引用预处理语句参数。
- 在视图定义中允许使用 order by，但是，如果从特定视图进行了选择，而该视图使用了具有自己 order by 的语句，它将被忽略。
- 在定义中引用的表或视图必须存在，但是，创建了视图后，能够舍弃定义引用的表或视图。要想检查视图定义是否存在这类问题，可使用 check table 语句。
- 在定义中不能引用 temporary 表，不能创建 temporary 视图。
- 不能将触发程序与视图关联在一起。

2. 在单表上创建视图

在 MySQL 中可以在单个表上创建视图。

【例 6-6】 在 teacher 表上创建一个简单的视图，视图名称为 teach_view1。

代码和运行结果如下：

```
mysql> create view teach_view1
    -> as select * from teacher;
       Query OK, 0 rows affected (0.70 sec)
```

可以利用 select 语句查询视图 teach_view1 的数据如下：

```
mysql> select * from teach_view1;
       +----------+-------+--------+-------+---------+
       | teacherno | tname | major | prof | department |
       +----------+-------+--------+-------+---------+
       | t05001 | 苏超然 | 软件工程 | 教授 | 计算机学院 |
       | t05002 | 常杉 | 会计学 | 助教 | 管理学院 |
       | t05003 | 孙释安 | 网络安全 | 教授 | 计算机学院 |
       | t05011 | 卢敖治 | 软件工程 | 副教授 | 计算机学院 |
       | t05017 | 茅佳峰 | 软件测试 | 讲师 | 计算机学院 |
       | t06011 | 夏南望 | 机械制造 | 教授 | 机械学院 |
       | t06023 | 葛庭宇 | 铸造工艺 | 副教授 | 材料学院 |
       | t07019 | 韩既乐 | 经济管理 | 讲师 | 管理学院 |
       | t08017 | 时观 | 金融管理 | 副教授 | 管理学院 |
       +----------+-------+--------+-------+---------+
       9 rows in set (0.25 sec)
```

3. 在多表上创建视图

MySQL 数据库中也可以在两个或两个以上的表上创建视图。

【例 6-7】 在 student 表、course 表和 score 表上创建一个名为 stu_score1 的视图。视图中保留 18 级的女生的学号、姓名、电话、课程名和期末成绩。

代码和运行结果如下：

```
mysql> create view stu_score1
```

```
    -> as select student.studentno, sname, phone, cname,final
    -> from score join student on student.studentno = score. studentno
    -> join course on course.courseno = score.courseno
    -> where sex = '女' and left(student.studentno,2) =  '18';
       Query OK, 0 rows affected (0.06 sec)
```

此视图保存三个表的数据,可以利用 select 语句查询视图 stu_score1 的数据如下:

```
mysql> select * from stu_score1;
       +------------+--------+------------+---------+-------+
       | studentno  | sname  | phone      | cname   | final |
       +------------+--------+------------+---------+-------+
       | 18122221324 | 何白露 | 13178978999 | 电子技术 | 62.0  |
       | 18122221324 | 何白露 | 13178978999 | C语言    | 77.0  |
       | 18125121107 | 梁一苇 | 13145678921 | 电子技术 | 86.0  |
       | 18125121107 | 梁一苇 | 13145678921 | C语言    | 60.0  |
       | 18135222201 | 凌浩风 | 15978945645 | C语言    | 92.0  |
       | 18135222201 | 凌浩风 | 15978945645 | 会计软件 | 82.0  |
       +------------+--------+------------+---------+-------+
       6 rows in set (0.09 sec)
```

4. 在已存在的视图上创建视图

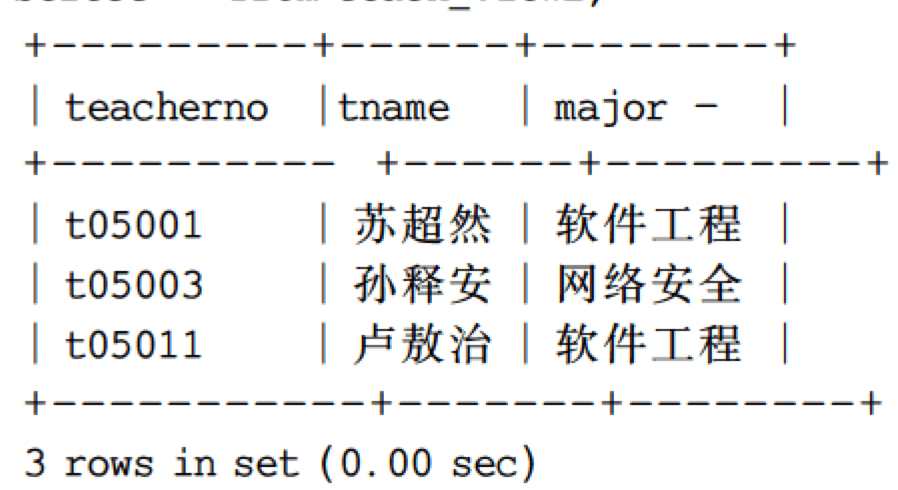
【例 6-8】 创建视图 teach_view2,统计计算机学院的教师中的教授和副教授的教师号、教师名和专业。

代码和运行结果如下:

```
mysql> create view teach_view2
    -> as select teacherno, tname, major
    -> from teach_view1
    -> where prof like '%教授' and department = '计算机学院';
       Query OK, 0 rows affected (0.08 sec)
```

可以通过用 select 语句查看视图 teach_view2 的数据如下:

```
mysql> select * from teach_view2;
```



```
       +----------+------+--------+
       | teacherno |tname  | major -  |
       +----------  +------+---------+
       | t05001    | 苏超然 | 软件工程 |
       | t05003    | 孙释安 | 网络安全 |
       | t05011    | 卢敖治 | 软件工程 |
       +-----------+-------+--------+
       3 rows in set (0.00 sec)
```

说明:

(1) 定义视图时基本表可以是当前数据库的表,也可以来自于另外一数据库的基本表,只要在表名前添加数据库名称即可,如 mysql. student02。

(2) 定义视图时可在视图名后面指明视图列的名称,名称之间用逗号分隔,但列数要与 select 语句检索的列数相等。例如,定义视图 teach_view2 可以写成如下方式:

```
create view teach_view2(教师号,教师名,专业)
as select teacherno, tname, major...
```

(3) 使用视图查询时,若其基本表中添加了新字段,则该视图将不包含新字段。

(4) 如果与视图相关联的表或视图被删除,则该视图将不能再使用。

6.2.2 查看视图的定义

查看视图是指查看数据库中已存在的视图的定义。查看视图必须要有 show view 的权限,MySQL 数据库下的 user 表中保存着这个信息。查看视图的方法包括 describe 语句、show table status 语句、show create view 语句和查询 information_schema 数据库下的 views 表等。

(1) 使用 describe 语句查看视图基本信息。可以使用 describe 语句查看表的基本定义。同样可以使用 describe 语句查看视图的基本定义。使用 describe 语句查看视图的基本形式与查看表的形式是一样的。

(2) 利用 show table status 语句查看视图的基本信息。MySQL 中,可以使用 show table status 语句来查看视图的信息。其语法形式如下:

```
show table status like 'view_name';
```

其中,like 表示后面匹配的是字符串;view_name 参数指要查看的视图的名称,需要用单引号引起。

(3) 利用 show create view 语句查看视图详细信息。MySQL 中,利用 show create view 语句可以查看视图的详细定义。其语法形式如下:

```
show create view view_name
```

(4) 在 views 表中查看视图的详细信息。MySQL 数据库中,所有视图的定义都存在 information_schema 数据库下的 views 表中。例如,查询 information_schema. views 表,可以查看到数据库中所有视图的详细信息。代码如下:

```
select * from information_schema.views;
```

其中,* 表示查询所有的列的信息;information_schema. views 表示 information_schema 数据库下面的 views 表。

6.2.3 修改视图


修改视图

修改视图是指修改数据库中已存在的表的定义。当基本表的某些字段发生改变时,可以通过修改视图来保持视图和基本表之间的一致。MySQL 中通过 create or replace view 语句和 alter 语句来修改视图。

MySQL 中,create or replace view 语句可以用来修改视图。该语句的使用非常灵活。在视图已经存在的情况下,对视图进行修改;视图不存在时,可以创建视图。

在 MySQL 中,alter 语句可以修改表的定义,可以创建索引。不仅如此,alter 语句还可以用来修改视图。alter 语句修改视图的语法格式如下:

```
alter [algorithm = {undefined|merge|temptable}]
view view_name [(column_list)]
```

```
as select 语句
[with [cascaded|local]check option];
```

【例 6-9】 修改视图 teach_view2，统计计算机学院和材料学院的教师中的教授和副教授的教师号、教师名和专业，并在视图名后面指明视图列名称。

代码和运行结果如下：

```
mysql > alter view teach_view2(教师号,教师名,专业)
    -> as select teacherno, tname, major
    -> from teach_view1
    -> where prof like '%教授'
    -> and (department = '计算机学院' or department = '材料学院');
     Query OK, 0 rows affected (0.04 sec)
```

可以通过 select 语句查看视图 teach_view2 的数据如下：

```
mysql > select * from teach_view2;
        +------+------+--------+
        | 教师号 | 教师名 | 专业     |
        +------+------+--------+
        | t05001 | 苏超然 | 软件工程 |
        | t05003 | 孙释安 | 网络安全 |
        | t05011 | 卢敖治 | 软件工程 |
        | t06023 | 葛庭宇 | 铸造工艺 |
        +------+------+--------+
        4 rows in set (0.00 sec)
```

6.2.4 利用 MySQL Workbench 工具创建和管理视图



利用 Workbench 创建视图

1. 利用 MySQL Workbench 创建视图

(1) 启动 MySQL Workbench 工具，单击实例 mysql57，选择当前数据库 teaching。

(2) 在 teaching 数据库中选择 views，展开 views 选项，可以看到已经创建的视图，在如图 6-11 所示的弹出菜单中，执行 Create View 命令。

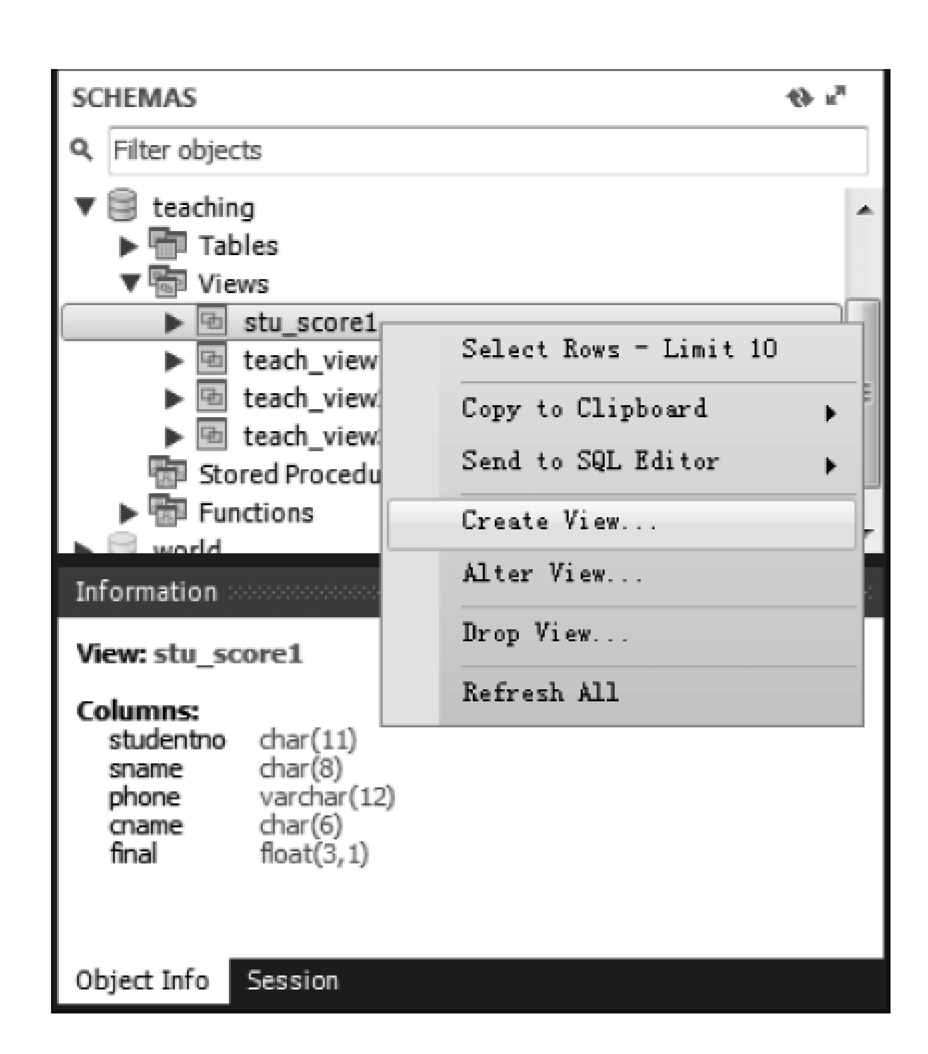


图 6-11 创建视图命令

(3) 在文本编辑区，按照如图 6-12 所示输入创建视图 stu_view1 的内容。实现以 student 表和 score 表为基表的视图中保留 19 级的男生的学号、姓名、电话、课程号和期末成绩的功能。

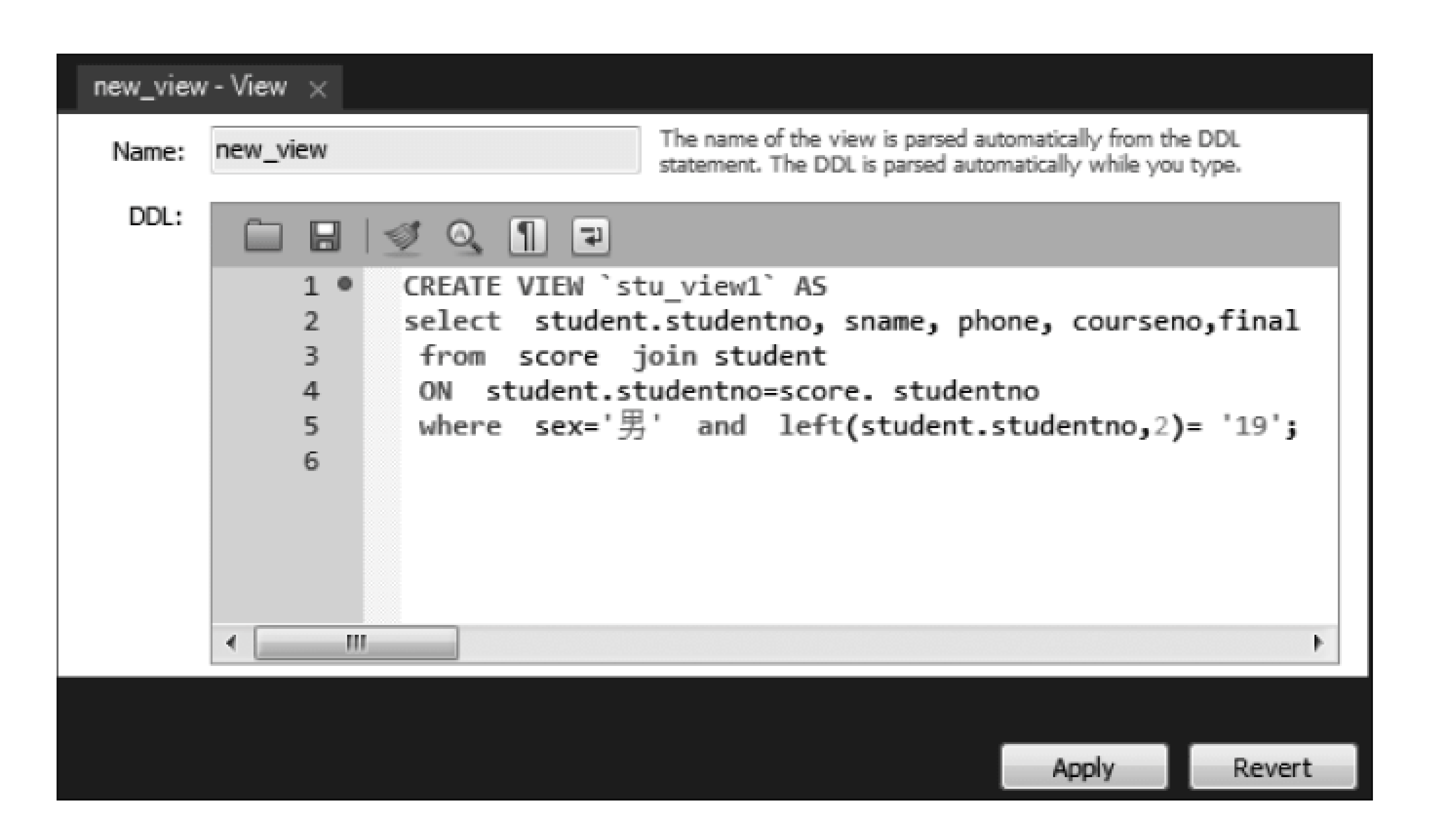


图 6-12　输入视图 stu_view1 的内容

(4) 自己检查无误后，单击 Apply 按钮，进入如图 6-13 所示的代码对话框中，这是要向数据库 teaching 中存储的脚本。

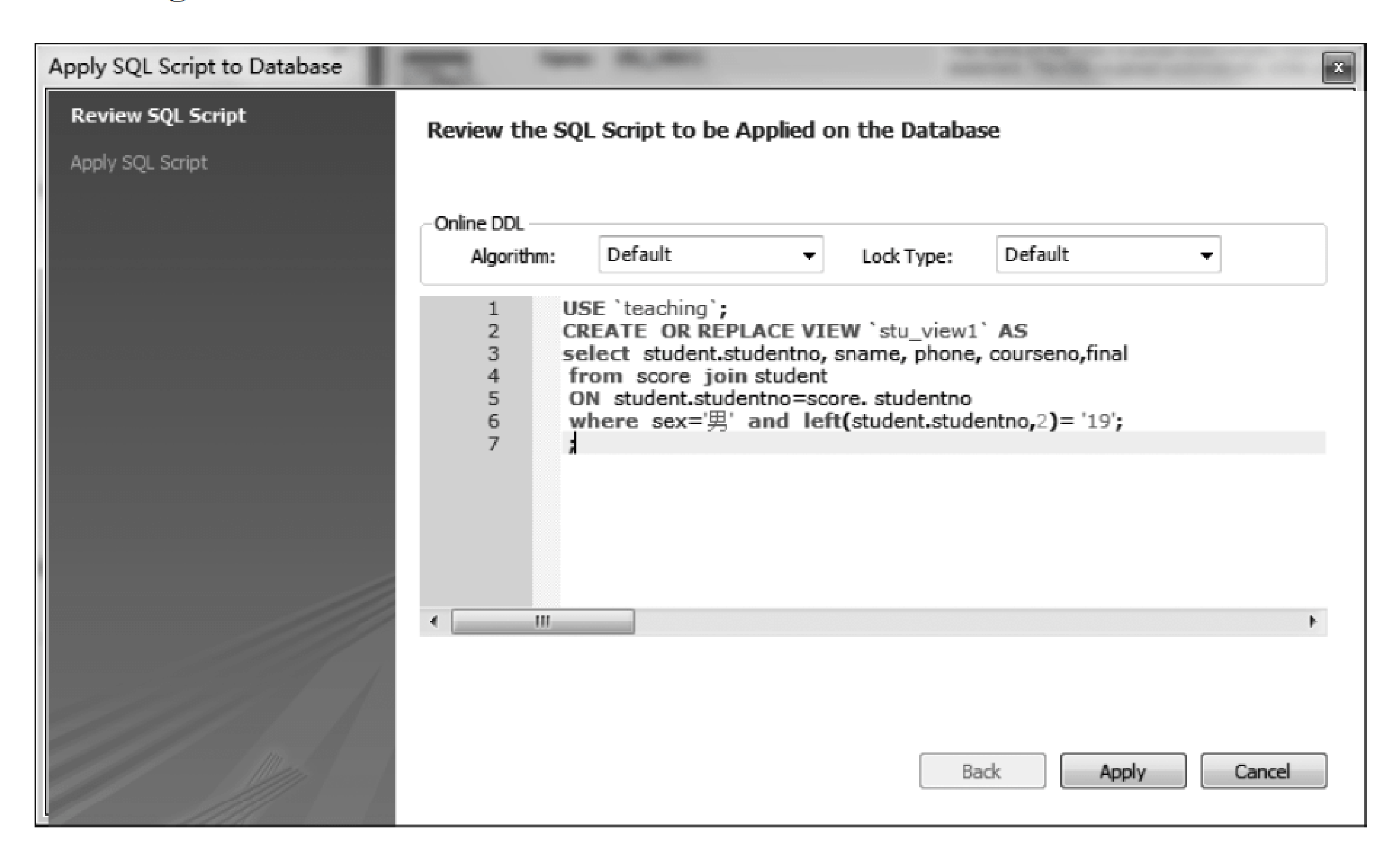


图 6-13　存储的脚本

(5) 单击 Apply 按钮，进入如图 6-14 所示的对话框中，可以通过 Show logs(Hide logs) 转换按钮查看信息记录(Massage Log)窗口中的信息。可以查看到成功创建视图的提示：SQL script was successfully applied to the database。单击 Finish 按钮完成视图创建过程。

(6) 在数据库 teaching 中展开 view 文件夹，找到视图 stu_view1，执行 Select Rows-Limit 10 命令，即可看到视图 stu_view1 的查询结果。即含有 19 级的男生的学号、姓名、电话、课程号和期末成绩的结果集，如图 6-15 所示。

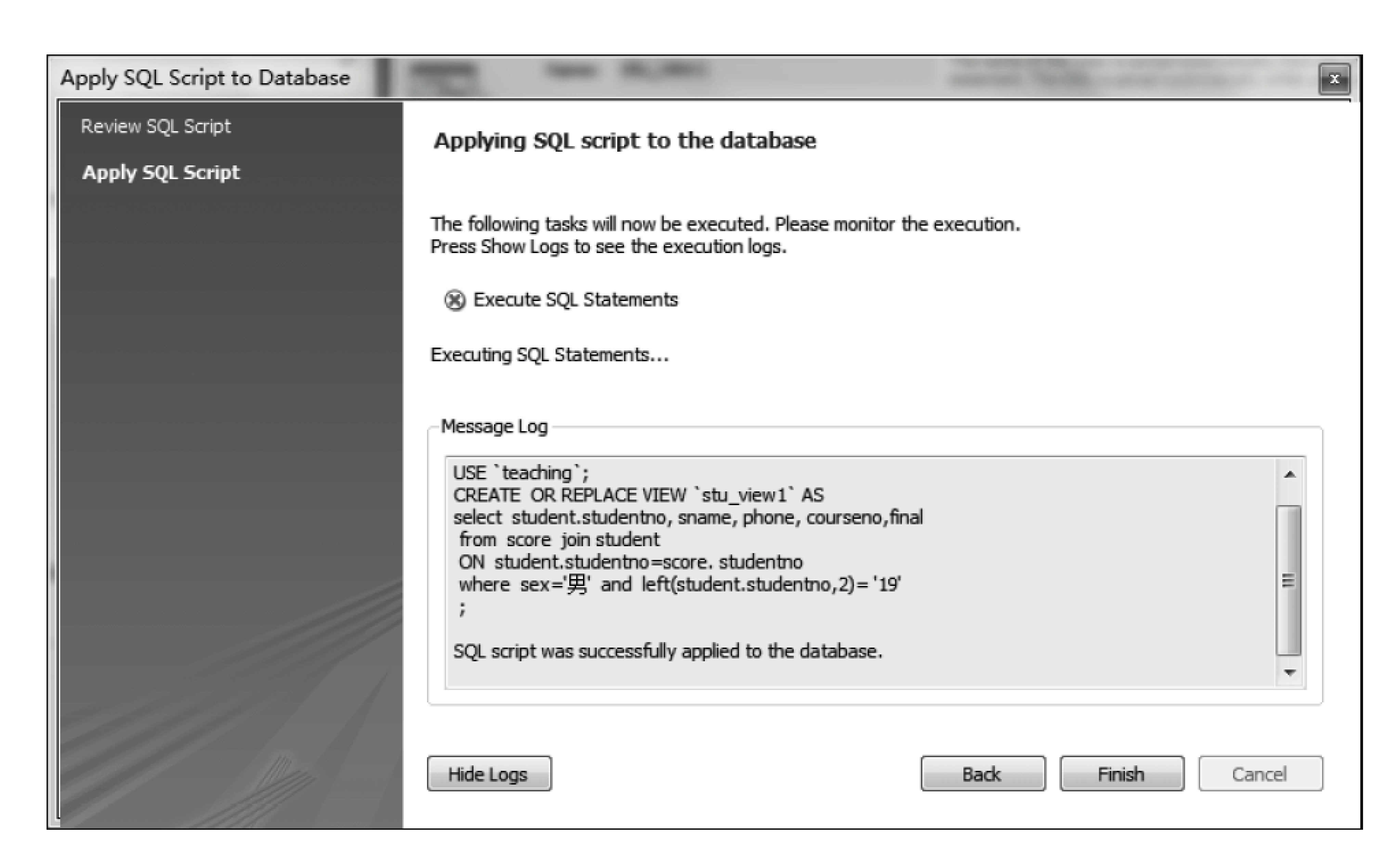


图 6-14　完成视图创建过程

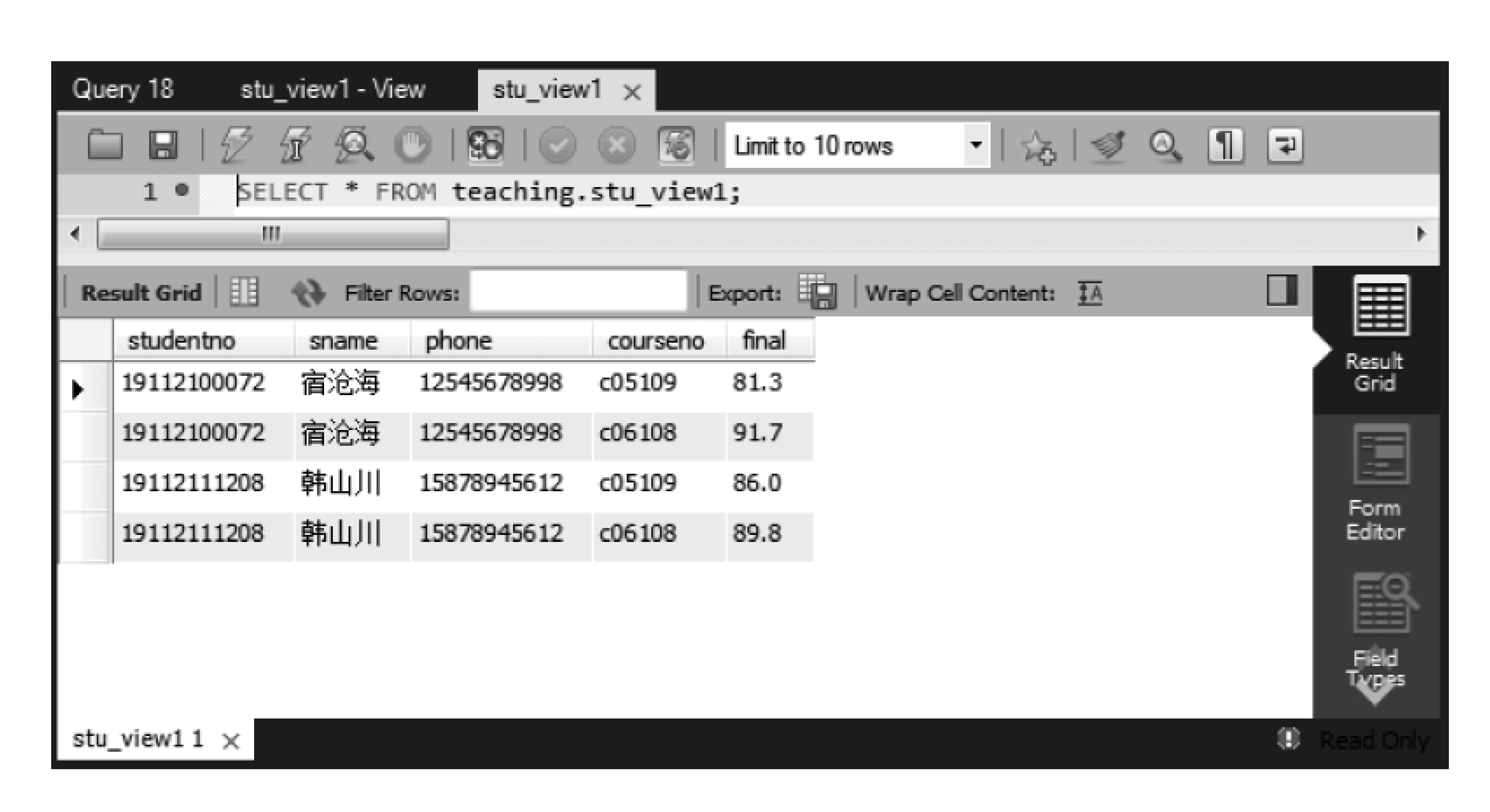


图 6-15　stu_view1 的查询结果

2. 利用 MySQL Workbench 修改视图



利用 Workbench 修改视图

(1) 在数据库 teaching 中展开文件夹，右击视图 stu_view1，执行 Alter View 命令，进入如图 6-16 所示的修改对话框中。

(2) 如图 6-17 所示。输入修改项，例如，视图名改为 stu_view2、各个视图列的输出名称改为汉语标识，过滤条件改为 18 级。

(3) 依次单击对话框中 Apply 按钮和 Finish 按钮即可完成视图修改。

(4) 在数据库 teaching 中展开 view 文件夹，找到视图 stu_view2，执行 Select Rows-Limit 10 命令，即可看到视图 stu_view2 的查询结果。即含有 18 级的男生的学号、姓名、电话、课程号和期末成绩的结果集，如图 6-18 所示。

图 6-16 修改视图

图 6-17 输入修改项

6.2.5 删除视图

删除视图是指删除数据库中已存在的视图。删除视图时，只能删除视图的定义，不会删除数据。MySQL 数据库中，用户必须拥有 drop 权限才能使用 drop view 语句来删除视图。

对需要删除的视图，使用 drop view 语句进行删除。使用 drop view 命令可以删除多个视图，各视图名之间用逗号分隔。基本格式如下：

```
drop view [if exists]viewname1[,...] [restrict|cascaded]
```

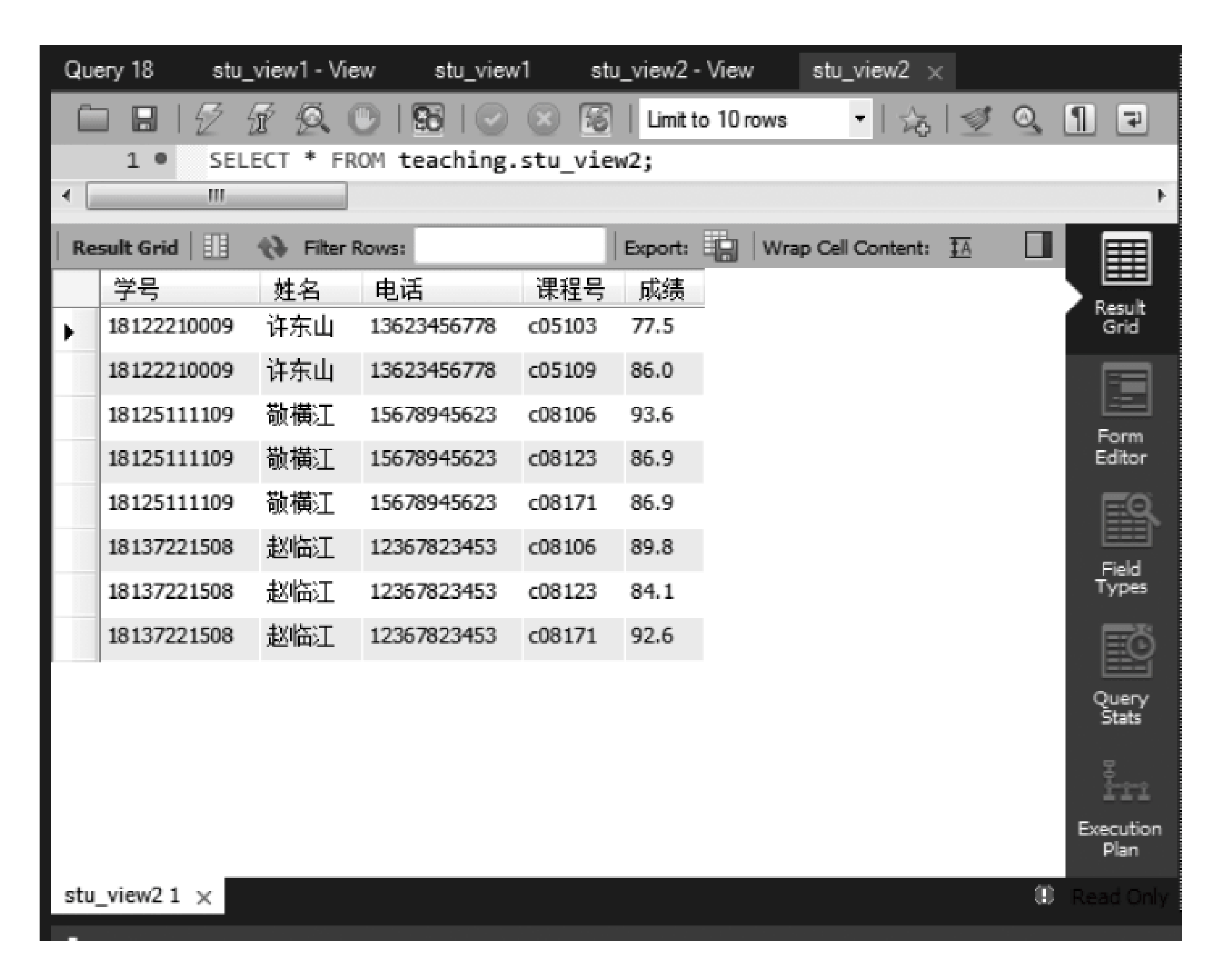


图 6-18　视图 stu_view2 的查询结果

例如，删除视图 V1_student 的命令如下：

```
drop view V1_student;
```

如果在 MySQL Workbench 中删除视图，只要右击要删除的视图，执行 Drop View 命令，按照操作提示就可以完成。

6.3　视图的应用

视图的使用主要包括视图的检索，以及通过视图对基表进行插入、修改、删除操作。视图的检索几乎没有什么限制，但是对通过视图实现表的插入、修改、删除操作则有一定的限制条件。

6.3.1　使用视图管理表数据



视图的应用

1. 使用视图进行查询

使用视图进行查询实际上就是把视图作为数据源，实现查询功能。

【例 6-10】 通过视图 stu_view2，查询选修课程号为 c08123、且成绩在 80 分以上的 18 级男生的学号、课程号和成绩。

代码和运行结果如下：

```
mysql> select 学号, 姓名,课程号,成绩
    -> from stu_view2
    -> where 课程号 = 'c08123' and 成绩 > 80;
      +-----------+------+------+-----+
      | 学号        | 姓名   | 课程号 | 成绩 |
      +-----------+------+------+-----+
```

```
| 18125111109  | 敬横江 | c08123 | 86.9  |
| 18137221508  | 赵临江 | c08123 | 84.1  |
+-------------+-------+-------+------+
2 rows in set (0.00 sec)
```

2. 使用视图进行统计计算

【例 6-11】 创建视图 course_avg,统计各门课程的平均成绩,并按课程名称降序排列。代码和运行结果如下:

```
mysql> create view course_avg
    -> as select cname 课程名, avg(final) 平均成绩
    -> from score join course on score.courseno = course.courseno
    -> group by cname desc;
       Query OK, 0 rows affected (0.05 sec)
mysql> select * from course_avg;
        +--------+--------+
        | 课程名   | 平均成绩 |
        +--------+--------+
        | 经济法   | 92.00000 |
        | 金融学   | 85.50000 |
        | 机械制图 | 86.66667 |
        | 机械设计 | 84.66667 |
        | 会计软件 | 84.25000 |
        | 电子技术 | 77.66667 |
        | C 语言   | 79.85714 |
        +--------+--------+
        7 rows in set (0.04 sec)
```

3. 使用视图修改基本表数据

使用视图修改表数据,是指在视图中进行 insert、update 和 delete 等操作而修改基表的数据。通过视图修改表数据时,要有执行相关操作的权限。

【例 6-12】 通过视图 teach_view1,对基表 teacher 进行插入、更新和删除数据的操作。代码和运行结果如下:

```
mysql> insert into teach_view1(teacherno,tname,major,prof,department)
    -> values ('t06027', '陶期年', '纳米技术', '教授', '材料学院');
        Query OK, 1 row affected (0.07 sec)
mysql> update teach_view1 set prof = '副教授' where teacherno = 't07019';
        Query OK, 1 row affected (0.07 sec)
        Rows matched: 1 Changed: 1 Warnings: 0
mysql> delete from teach_view1 where teacherno = 't08017';
        Query OK, 1 row affected (0.04 sec)
```

使用 select 语句查询 teacher 表,可以看到基表中的数据也相应地进行了修改。

```
mysql> select * from teacher ;
        +----------+------+--------+-----+---------+
        | teacherno | tname | major    | prof | department |
        +----------+------+--------+-----+---------+
        | t05001    | 苏超然 | 软件工程 | 教授  | 计算机学院 |
```

```
| t05002     | 常杉   | 会计学   | 助教   | 管理学院   |
| t05003     | 孙释安 | 网络安全 | 教授   | 计算机学院 |
| t05011     | 卢敖治 | 软件工程 | 副教授 | 计算机学院 |
| t05017     | 茅佳峰 | 软件测试 | 讲师   | 计算机学院 |
| t06011     | 夏南望 | 机械制造 | 教授   | 机械学院   |
| t06023     | 葛庭宇 | 铸造工艺 | 副教授 | 材料学院   |
| t06027     | 陶期年 | 纳米技术 | 教授   | 材料学院   |
| t07019     | 韩既乐 | 经济管理 | 副教授 | 管理学院   |
+----------+------+--------+------+---------+
9 rows in set (0.00 sec)
```

【例 6-13】 视图 stu_score1 依赖于源表 student、course 和 score 三张表，包括 studentno、sname、phone、cname 和 final 5 个字段，通过 stu_score1 修改基本表 student 中的学号为 18125121107 的电话号码。

代码和运行结果如下：

```
mysql > update stu_score1 set phone = '132123456777'
    -> where studentno  = '18125121107';
      Query OK, 1 row affected (0.12 sec)
      Rows matched: 1 Changed: 1 Warnings: 0
```

通过查看 student 表，可以看到相应成绩已做了更改。

```
mysql > select studentno, sname, phone from student
    -> where studentno  = '18125121107' ;
      +-----------+-------+-----------+
      | studentno   | sname  | phone      |
      +-----------+-------+-----------+
      | 18125121107 | 梁一苇  | 13212345677 |
      +-----------+-------+-----------+
      1 row in set (0.01 sec)
```

说明：

(1) 视图若只依赖于一个基表，则可以直接通过视图来更新基本表的数据。

(2) 若一个视图依赖于多张基表，则一次只能修改一个基表的数据，不能同时修改多个基表的数据。

(3) 如果视图包含下述结构中的任何一种，都是不可修改的：

- 视图的列含有聚合函数。
- 视图的列是通过表达式并使用列计算出其他列。
- 含有 distinct 关键字。
- 含有 group by 子句、order by 子句、having 子句。
- 含有 union 运算符。
- 视图的列位于选择列表中的子查询。
- from 子句中包含多个表。
- select 语句中引用了不可更新视图。
- where 子句中的子查询，引用 from 子句中的表。

6.3.2 检查视图的应用



检查视图的应用

在 MySQL 数据库中,视图可分为普通视图与检查视图。前面介绍的视图都没有使用 with check option 子句,当没有 with_check_option 时,表示 with_check_option 的值为 0。即为普通视图,普通视图不具备检查功能。如果使用了 with check option 子句,在通过检查视图更新基表数据时,只有满足检查条件的更新语句才能成功执行。

【例 6-14】 编程在 teaching 数据库中创建一个名称为 V_ dept 的视图,包含所有部门为"计算机学院"的老师的数据信息,需限制插入数据中部门必须为"计算机学院"。

分析:该程序通过单表生成的视图 V_dept 向基表 teacher 中插入一条记录,并通过查询语句显示基表中的所有数据。

代码和运行结果如下:

```
-- 在"查询编辑器"中输入以下程序,创建 V_dept 视图
mysql> create view V_dept
    -> AS
    -> select teacherno,tname,major,prof, department
    -> from teacher
    -> where department  = '计算机学院'
    -> with check option;
       Query OK, 0 rows affected (0.05 sec)
    -- 通过视图 V_ dept 向基表 teacher 中插入数据
mysql> insert into V_dept
    -> values('t08017','时观','金融管理','副教授','计算机学院');
       Query OK, 1 row affected (0.06 sec)
mysql> select * from teacher where tname = '时观';
       +---------+-------+--------+--------+----------+
       | teacherno | tname  | major    | prof    | department |
       +---------+-------+--------+--------+----------+
       | t08017    | 时观   | 金融管理 | 副教授  | 计算机学院 |
       +---------+-------+--------+--------+----------+
       1 row in set (0.00 sec)
```

本例由于创建了 with check option 检查条件约束,当插入记录时所有"部门"信息不符合条件的记录无法插入和修改,并显示错误提示信息。

```
-- 通过视图 V_sex 向基表 teacher 中插入数据行('t08037','时刻','软件技术','讲师','软件学院')
mysql> insert into V_dept
     -> values('t08037','时刻','软件技术','讲师','软件学院');
ERROR 1369 (HY000): check option failed 'teaching.v_dept'
```

执行结果表明,通过检查更新表数据时,检查视图对更新数据进行了先行检查,若更新语句的数据不满足检查条件,则检查视图就会抛出异常,更新失败。

另外,检查视图又可以分为 local 视图与 cascade 视图。当 with_check_option 的值为 1 时表示 local 视图,值为 2 时表示 cascade 视图。cascade 视图又称级联视图,是在视图的基础上再次创建另一个视图。相关内容感兴趣的读者可以进一步学习探讨。

6.4 小　结

本章介绍了 MySQL 数据库的创建和管理索引的基础知识,创建索引的方法、删除索引的方法。还介绍了视图的定义、视图的作用、创建视图、删除视图、查询视图和更新视图等内容。视图一经定义,就可以像基表一样进行查询。用户还可以利用视图修改基表数据,但有一定的限制。

学会利用 MySQL Workbench 工具创建和管理数据库对象,如视图和索引,对于初学者理解和掌握 MySQL 中的概念和基本操作命令,能够起到不可替代的作用。学习本章后,需要重点掌握如下内容:

- 索引、视图的作用和用途。
- 索引、视图的创建、管理和删除方法。
- 索引、视图的常用命令。
- 利用视图对数据表的数据进行修改操作。

习　题　6

1. 选择题

(1) 下列________语句不能用于创建索引。

A. create index　　B. create table

C. alter table　　D. create database

(2) 下面对索引的相关描述正确的是________。

A. 经常被查询的列不适合建索引　　B. 小型表适合建索引

C. 有很多重复值的列不适合建索引　　D. 是外键或主键的列不适合建索引

(3) MySQL 中不可对视图执行的操作有________。

A. select　　B. insert　　C. delete　　D. create index

(4) 对视图的描述错误的是________。

A. 视图是一张虚拟表

B. 视图定义包含 limit 子句时才能设置排序规则

C. 可以像查询表一样来查询视图

D. 被修改数据的视图只能是一个基表的列

(5) with check option 属性对视图有________的用途。

A. 进行权限检查　　B. 进行删除监测　　C. 进行更新监测　　D. 进行插入监测

(6) 索引可以提高________操作的效率。

A. insert　　B. update　　C. delete　　D. select

(7) 在 MySQL 中唯一索引的关键字是________。

A. fulltext　　B. only　　C. unique　　D. index

2. 思考题

(1) 简述创建索引的必要性。

（2）MySQL 中普通索引、主键索引和唯一性索引的区别是什么？

（3）简述表和视图之间的关系。

（4）简述创建视图的必要性。

3. 上机练习题（本题利用 teaching 数据库中的表进行操作）

（1）在 course 表的 cname 列上创建索引 IDX_cname。

（2）在 student 表的 studentno 和 phone 列上创建唯一性索引 uq_stu。并输出 student 表中的记录，查看输出结果的顺序。

（3）创建一个视图 v_teacher，查询所有“计算机学院”教师的信息。

（4）创建一个视图 v_avgstu，查询每个学生的学号、姓名及平均分，并且按照平均分降序排序。

（5）修改 v_teacher 的视图定义，添加 with check option 选项。

（6）通过视图 v_teacher 向基表 teacher 中分别插入下列数据，并查看插入数据的情况。

```
('t05039','张馨月','计算机应用', '讲师', '计算机学院)
('t06018', '李书诚', '机械制造','副教授','机械学院')
```

（7）通过视图 v_teacher 将基表 teacher 中教师编号为 t05039 的教师职称修改为“副教授”。

第 7 章 MySQL 编程基础

在前面介绍的控制台模式下，MySQL 客户端可以执行单句的 MySQL 命令，执行数据表的 SQL 语句进行信息查询等。如果要完成较复杂的操作，就需要一次执行一系列的命令，这就需要利用 MySQL 的脚本来实现。MySQL 的脚本就是通常说的 MySQL 程序，是通过一套对字符、关键词以及特殊符号的使用规定，利用一条或多条 MySQL 语句（SQL 语句＋扩展语句）编写而成的。MySQL 的脚本文件保存时后缀名一般为. sql。

MySQL 脚本具体来说是由常量、变量、函数、表达式、关键词等组成的语句，外加注释构成的。MySQL 语句是组成 MySQL 脚本的基本单位，每条语句能完成特定的操作。

MySQL 程序包含三种基本结构即顺序结构、选择结构和循环结构。实现这三种基本结构的语句是 MySQL 中的控制流语句。

本章主要介绍利用 MySQL 语言进行数据库编程的基础知识，以及自定义函数的创建和应用等内容。

7.1 MySQL 编程基础知识

7.1.1 自定义变量的应用

MySQL 的每一个客户机成功连接服务器后，都会产生与之对应的会话。会话期间，MySQL 服务实例会在 MySQL 服务器内存中生成与该会话对应的会话系统变量，这些会话系统变量的初始值是全局系统变量值的拷贝。除此之外，MySQL 的用户还可以利用自己定义的变量。



自定义变量

MySQL 语言中的自定义变量由变量名、变量类型和变量值三要素构成。变量名要求是标识符，不能与关键词和函数名相同。变量类型和常量类型一样，决定变量存储空间和取值范围，变量值要求符合本类型取值范围的要求。计算机中的变量和在数学中所遇到的变量的概念基本一样，可以随时改变它所对应的数值，但需要实现存储要求。

用户自定义变量在 MySQL 系统中，也存在两种类型，即以@开头的用户会话变量和局部变量。

1. 用户会话变量

系统会话变量与用户会话变量的共同之处在于：变量名大小写不敏感。系统会话变量与用户会话变量的区别在于：

- 用户会话变量一般以一个“@”开头；系统会话变量以两个“@”开头。

• 系统会话变量无须定义可以直接使用。

(1) 用户会话变量的使用过程。一个用户会话变量创建之后,就可以作为表达式或表达式的组成因素用于其他 SQL 语句中,如图 7-1 所示。MySQL 客户机 A 定义了会话变量,会话期间,该会话变量一直有效;MySQL 客户机 C 不能访问客户机 A 定义的会话变量;客户机 A 关闭或者客户机 A 与服务器断开连接后,客户机 A 定义的所有会话变量将自动释放,以便节省 MySQL 服务器的内存空间。同样,客户机 C 中定义的会话变量也是如此。

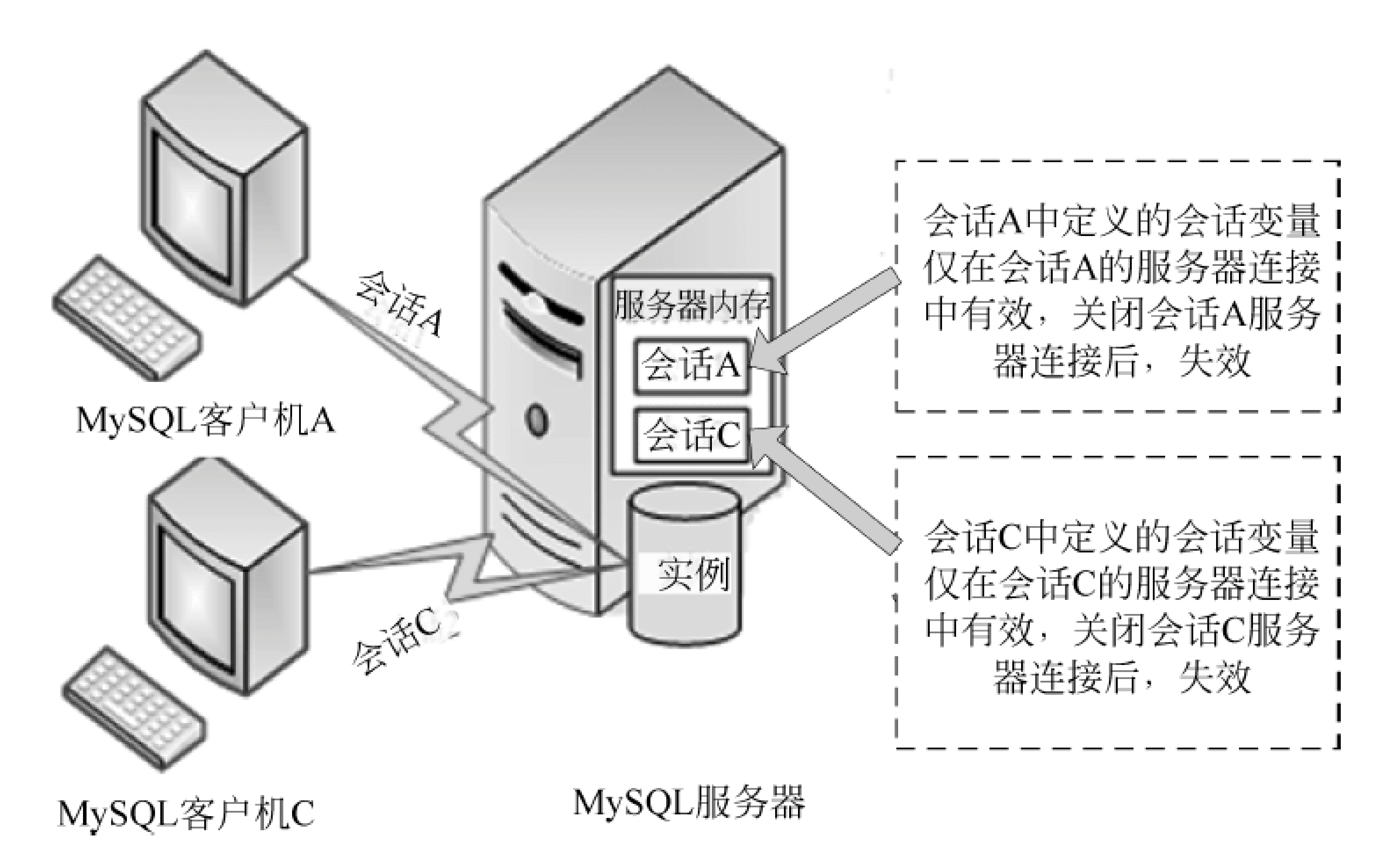


图 7-1 用户会话变量的使用过程

实际上是 MySQL 服务器在内存中为每一个会话开辟独立的会话连接空间,不同的会话空间互不干扰,会话结束,会话空间释放。而会话变量的生存期就是所在会话空间开辟到释放的这一段时间。

(2) 用户会话变量的定义与赋值。一般情况下,用户会话变量的定义与赋值会同时进行。定义和初始化一个用户会话变量可以使用 set 或 select 语句。

方法 1:使用 set 命令定义用户会话变量,并为其赋值,语法格式如下:

```
set @user_variable1 = expression1 [,@user_variable2 = expression2 , … ]
```

方法 2:使用 select 语句定义用户会话变量,并为其赋值,语法格式有两种。

```
select @user_variable1 := expression1 [,user_variable2 := expression2 , … ]
```

或

```
select expression1 into @user_variable1, expression2 into @user_variable2,…
```

说明:

① 用户会话变量的数据类型是根据赋值运算符“=”右边表达式的计算结果自动分配的。也就是说,等号右边的值(包括字符集和字符序)决定了用户会话变量的数据类型(包括字符集和字符序)。

② 使用 select 语句定义用户会话变量时,赋值号采用“:=”形式,能够产生结果集。而利用 into 赋值的方式的 select 语句,仅仅用于会话变量的定义及赋值,但不会产生结果集。

例如:创建用户会话变量@name 并赋值为“赵临江”。可以用如下两种命令实现。

```
set @name = '赵临江'
```

或:

```
select @name := '赵临江'
```

③ 赋值号":="与"="的总结比较:":="是赋值号,能够实现赋值操作,即将右边的值赋值给左边的变量。"="一般情况下是作为比较操作符使用的。特殊情况下,"="则只在 set 语句里面作为赋值号使用,包括 update 语句里面的 set 子句。如"set @var1=value;"。

【例 7-1】 使用查询结果给变量赋值。

代码和运行结果如下:

```
mysql> use teaching;
       Database changed
mysql> set @sname = (select sname from student
    -> where studentno = '19126113307');
       Query OK, 0 rows affected (0.00 sec)
mysql> select studentno, sname, birthdate
    -> from student where sname = @sname;
       +-----------+------+-----------+
       | studentno   | sname | birthdate   |
       +-----------+------+-----------+
       | 19126113307 | 梅惟江 | 2003-09-07 |
       +-----------+------+-----------+
       1 row in set (0.00 sec)
```

【例 7-2】 利用 select 语句将表中数据赋值给变量。

命令和运行结果如下:

```
mysql> select @sname := sname from student limit 0,1;
       +--------------+
       | @sname := sname |
       +--------------+
       | 许东山          |
       +--------------+
       1 row in set (0.03 sec)
```

2. 局部变量

局部变量是指在其定义的某个局部程序范围内有效的变量。

(1) 局部变量的定义与赋值。declare 命令专门用于定义局部变量及对应的数据类型。

例如,定义局部变量 myvar,数据类型为 int,默认值为 100,代码如下:

```
declare myvar int default 100;
```

下面给局部变量 myvar 赋值为 77,代码如下:

```
set myvar = 77;
```

(2) 局部变量的使用。局部变量必须定义在函数、触发器、存储过程等存储程序中,局部变量的作用范围仅仅局限于存储程序中。局部变量主要用于下面三种场合:

- 局部变量必须先定义,才可以使用 set 命令或者 select 语句为其赋值。局部变量定义在 begin…end 语句块之间。此时局部变量首先必须使用 declare 命令定义,并且必须指定局部变量的数据类型。
- 局部变量作为存储过程或者函数的参数使用。此时虽然不需要使用 declare 命令定义,但需要指定参数的数据类型。
- 在 SQL 语句中使用局部变量。数据检索时,如果 select 语句的结果集是单个值,可以将 select 语句的返回结果赋予局部变量,局部变量也可以直接嵌入到 select、insert、update 以及 delete 语句的条件表达式中。

3. 局部变量与用户会话变量的区别

(1) 用户会话变量使用 set 命令或 select 语句定义并进行赋值,定义用户会话变量时无须指定数据类型。诸如“declare @student_no int;”的语句是错误语句,用户会话变量不能使用 declare 命令定义。

(2) 用户会话变量的作用范围与生存周期大于局部变量。用户会话变量在本次会话期间一直有效,直至关闭服务器连接。而局部变量如果作为存储过程或者函数的参数,此时在整个存储过程或函数内有效;如果定义在存储程序的 begin...end 语句块中,此时仅在当前的 begin...end 语句块中有效。

(3) 如果局部变量嵌入到 SQL 语句中,由于局部变量名前没有“@”符号,这就要求局部变量名不能与表字段名同名,否则将出现无法预期的结果。

在 MySQL 数据库中,由于局部变量涉及 begin...end 语句块、函数、存储过程等知识,局部变量的具体使用方法将结合这些知识在后面内容中进行讲解。

7.1.2 MySQL 表达式

MySQL 表达式是由运算符将常量、变量、字段名和函数等组合连接而成的有意义的字符序列。一个表达式通常可以得到一个值。具体来说,根据分类标准不同,可以对 MySQL 表达式进行不同的分类。

(1) 按照表达式值类型分类。与常量和变量一样,表达式的值也具有某种数据类型,可能的数据类型有字符类型、数值类型、日期时间类型。这样,根据表达式的值的类型,表达式可分为字符型表达式、数值型表达式和日期型表达式。

(2) 按照值形式分类。在 MySQL 语言中,当表达式的结果只是一个值,如一个数值、一个字符串或一个日期,这种表达式叫作标量表达式。例如:1+2,'a'>'b'。当表达式的结果是由不同类型的数据组成的一行值时,这种表达式叫作行表达式。例如,('18110123456','王达田','计算机',500)。当表达式的结果为 0 个、1 个或多个行表达式的集合时,那么这个表达式就叫作表表达式。

(3) 按照表达式形式分类。表达式还可分为单一表达式和复合表达式。单一表达式就是一个单一的值,如一个常量、变量、函数或列名。复合表达式是由运算符将多个单一表达式连接而成的表达式,例如:

```
1 + 7 + 3, a = v + 3, '2018 - 01 - 20' +  interval 6 month
```

7.1.3 定界符 delimiter 和 begin...end 语句块

前面在 MySQL 命令行客户端上执行的 MySQL 命令都是单条命令，为了解决更复杂的问题，MySQL 语言提供了自定义函数、存储过程等存储程序。在这些存储程序中，往往需要多条 SQL 命令或 MySQL 语句组合到一起执行。MySQL 语言可以利用 begin…end 语句块和重新设置定界符 delimiter 来实现。

1. 更改命令结束标记 delimiter


定界符 delimiter

默认 MySQL 的命令行结束符就是"；"，而函数和存储过程这样的语句中包含了很多的"；"，当创建函数或存储过程的时候就会报错。

默认情况下，不可能等到用户把这些语句全部输入完之后，再执行整段语句。因为 MySQL 一遇到分号，它就要自动执行。即在执行语句"return；"时，MySQL 数据库解释器就要执行了。这种情况下，就需要事先把 delimiter 换成其他符号，如//或＄＄。

为了避免 begin…end 语句块中的多条 MySQL 表达式被拆开，需要重置 MySQL 客户机中的命令结束标记（delimiter）。

利用 delimiter 定界符命令，可以重新定义一个语句执行的结束符。有 delimiter 定义的新定界符（即重置命令结束标记，如//或＄＄）就是告诉 MySQL 解释器，该段命令的结束和执行有了新的标识。

【例 7-3】 改变 MySQL 命令的结束标记示例。

代码和运行结果如下：

```
mysql > delimiter //
mysql > select studentno, sname, phone
    -> from student where sname like '赵 % '//
     +-----------+------+----------+
     | studentno   | sname | phone       |
     +-----------+------+----------+
     | 18137221508 | 赵临江 | 12367823453 |
     | 19123567897 | 赵既白 | 13175689345 |
     +-----------+------+----------+
     2 rows in set (0.03 sec)
mysql > delimiter  $ $
mysql > select studentno, sname, birthdate
     -> from student where sname like '梅 % '$ $
     +-----------+------+----------+
     | studentno   | sname | birthdate   |
     +-----------+------+----------+
     | 19126113307 | 梅惟江 | 2003 - 09 - 07|
     +-----------+------+----------+
     1 row in set (0.00 sec)
mysql > delimiter ;
```

其中 delimiter 定义好结束符为"＄＄"和"//"。最后又定义为"；"，是因为这时 MySQL 的默认结束符为"；"，实际编程时要养成习惯。

2. begin...end 语句块

通常利用 begin...end 可以用于定义一组语句块，在其他各大数据库中的客户端工具中

可直接调用，但在 MySQL 语言中不可以直接用。在 MySQL 语言中，局部变量、begin…end 语句块和流程控制语句等只能用于函数、存储过程、游标和触发器的定义内部。begin…end 语句块的简单形式如下：

```
begin
    [局部]变量声明;
    程序代码行集;
end ;                                    // end 之后以";"结束
```

7.1.4 预处理 SQL 语句


预处理 SQL 语句

前面介绍的 MySQL 语句在运行期间，SQL 语句不能发生动态的变化，这种 SQL 语句称为静态 SQL 语句。对于静态 SQL 语句而言，每次将其发送到 MySQL 服务实例时，MySQL 服务实例都会对其进行解析、执行，然后将执行结果返回给 MySQL 客户机。

MySQL 数据库还可以使用预处理的方式执行 SQL 语句。该类语句在运行期间，如果 SQL 语句或 SQL 所带的参数可以发生动态变化，这种 SQL 语句称为动态 SQL 语句或者预处理 SQL 语句。对于预处理 SQL 语句而言，预处理 SQL 语句创建后，第一次运行预处理 SQL 语句时，MySQL 服务实例会对其解析，解析成功后，将其保存到 MySQL 服务器缓存中，为今后每一次的执行做好准备(以后无须再次解析)。

这样就可以将某些 SQL 语句封装为预处理 SQL 语句，实现其“一次解析，多次执行”的性能优势。

1. 预处理 SQL 语句的格式

MySQL 数据库中的 prepare、execute、deallocate 统称为预处理语句(prepare statement)。一般格式如下：

```
prepare stmt_name from preparable_stmt;
execute stmt_name [using @var_name [, @var_name] ...];
{deallocate|drop} prepare stmt_name;
```

说明：

(1) prepare 语句用于预备一个语句，并赋予它名称 stmt_name，借此在以后引用该语句。preparable_stmt 可以是一个文字字符串或一个包含了语句文本的用户会话变量。该文本必须展现一个单一的 SQL 语句，而不是多个语句。使用本语句，“?”字符可以被用于制作参数，执行查询时，数据值在那里与查询结合在一起。参数制作符只能用于数据值应该出现的地方，不用于 SQL 关键词和标识符等。

如果带有此名称的预处理语句已经存在，则在新的语言被预备以前，它会被隐含地解除分配。这意味着，如果新语句包含一个错误并且不能被预备，则会返回一个错误，并且不存在带有给定名称的语句。

预处理语句的范围是客户端会话。在此会话内，语句被创建。其他客户端看不到它。

(2) execute 语句用于执行预处理语句。如果预处理语句包含任何参数制造符，则必须提供一个列举了用户会话变量(其中包含要与参数结合的值)的 using 子句。参数值只能由用户会话变量提供，using 子句必须准确地指明用户会话变量。用户会话变量的数目与

SQL 语句中的参数制造符的数量一样多。

若需要多次执行一个给定的预处理语句，在每次执行前，把不同的变量传递给它，或把变量设置为不同的值。

(3) deallocate prepare 语句用于释放预处理语句。如果终止一个客户端会话，同时没有对以前已预制的语句解除分配，则服务器会自动解除分配。

2. 预处理 SQL 语句的使用步骤

MySQL 支持预处理 SQL 语句，预处理 SQL 语句的使用主要包含三个步骤。创建预处理 SQL 语句、执行预处理 SQL 语句以及释放预处理 SQL 语句。

(1) 创建预处理 SQL 语句。

(2) 执行预处理 SQL 语句。

(3) 释放预处理 SQL 语句。

3. 预处理 SQL 语句的应用

【例 7-4】 在给定了两个直角边的长度时，计算直角三角形的斜边长度。

分析：可以通过使用文字字符串来创建一个预处理语句，以提供语句的文本，也可以将提供语句的文本赋值给一个用户会话变量。

代码和运行结果如下：

```
mysql> prepare hypo_c from 'select sqrt(pow(?,2) + pow(?,2)) AS hypotenuse';
       Query OK, 0 rows affected (0.00 sec)
       Statement prepared
mysql> set @a = 6;
       Query OK, 0 rows affected (0.00 sec)
mysql> set @b = 8;
       Query OK, 0 rows affected (0.00 sec)
mysql> execute hypo_c using @a, @b;
       +-----------+
       | hypotenuse |
       +-----------+
       |        10 |
       +-----------+
       1 row in set (0.00 sec)
mysql> deallocate prepare hypo_c;
       Query OK, 0 rows affected (0.00 sec)
```

【例 7-5】 利用预处理 SQL 语句输出 student 中的前两行记录的部分数据。

分析：利用预处理语句，可以使用 limit 子句指定记录行数。

代码和运行结果如下：

```
mysql> set @a = 2;
       Query OK, 0 rows affected (0.00 sec)
mysql> prepare STMT
    -> from "select studentno, sname, entrance from student limit ?";
       Query OK, 0 rows affected (0.00 sec)
       Statement prepared
mysql> execute STMT using @a;
       +-----------+-------+---------+
       | studentno  | sname | entrance |
```

```
+-----------+-------+---------+
| 18122210009 | 许东山 |    789   |
| 18122221324 | 何白露 |    879   |
+-----------+-------+---------+
2 rows in set (0.00 sec)
```

如果执行下列代码,则可以输出前三行记录,由此可以看出预处理 SQL 语句的作用。

```
mysql > set @a = 3;
        Query OK, 0 rows affected (0.00 sec)
mysql > execute STMT using @a;
        +-----------+-------+---------+
        | studentno   | sname  | entrance |
        +-----------+-------+---------+
        | 18122210009 | 许东山 |    789   |
        | 18122221324 | 何白露 |    879   |
        | 18125111109 | 敬横江 |    789   |
        +-----------+-------+---------+
        3 rows in set (0.00 sec)
```

说明:

(1) 使用预处理语句时,最好在编代码前,先测试预处理语句在应用程序中的运行情况。

(2) 预处理语句的 SQL 语法不能用于嵌套。也就是说,传递给 prepare 的语句本身不能是一个 prepare, execute 或 deallocate prepare 语句。

(3) 预处理语句的 SQL 语法与使用预处理语句 API 调用不同。例如,不能使用 API 函数来预备一个 prepare, execute 或 deallocate prepare 语句。

(4) 能够支持预处理操作的 SQL 语句如下: create table, delete, do, insert, replace, select, set, update 和多数的 show 语句,而不支持其他语句。

(5) 预处理语句的 SQL 语法可以在已存储的过程中使用,但是不能在已存储的函数或触发程序中使用。

7.1.5 注释

注释是程序代码中不被执行的文本字符串,用于对代码进行说明或进行诊断的部分语句。

(1) #(井号字符):从该字符到行尾都是注释内容。

(2) --(双连线字符):从双连线字符到行尾都是注释内容。注意,双连线后一定要加一个空格。

(3) 正斜杠星号字符(/ * … * /):开始注释对(/ *)和结束注释对(* /)之间的所有内容均视为注释。

例如,下面的程序代码中包含注释符号。

```
use teaching;                    -- 打开数据库
#查看学生的所有信息
select * from student;
/* 查看所有女生的学号、姓名、电话
```

```
附加条件是女生 */
select studentno,sname,phone from student
WHERE sex = '女';
```

7.2 自定义函数

第 2 章介绍了 MySQL 语言中常用的系统函数,实际上用户还可以根据自己业务的需要创建自定义函数。在 MySQL 中,可以利用 create function 语句创建自定义函数。创建函数必须具有 create routine 权限,并且 alter routine 和 execute 权限被自动授予它的创建者。需要注意的是,MySQL 的自定义函数定义时,需要指定当前数据库。



创建自定义函数

7.2.1 创建和调用自定义函数

1. 创建自定义函数的语法格式

在 MySQL 中,创建存储函数的基本语法如下:

```
create function func_name([[in | out | inout]func_parameter type[,…]])
returns return_type
[characteristic…]
begin
    function_body_statements;
    return[return_values];
end;
```

函数定义说明:

(1) create function:创建自定义函数的关键字。

(2) func_name:创建自定义函数的函数名。

(3) [in | out | inout] func_parameter type:函数参数形式。in 表示输入参数,out 表示输出参数,inout 表示输入输出参数,func_parameter 表示参数名,type 表示参数类型。

(4) returns return_type:函数返回值类型。

(5) characteristics:用于指定函数的特征参数,characteristics(函数选项)由以下一种或几种选项组合而成。

```
language sql
|[not] deterministic
|{ contains sql|no sql|reads sql data|modifies sql data }
|sql security {definer|invoker }
|comment 'string'
```

函数定义的 characteristics 选项说明:

- language sql:默认选项,用于说明函数体使用 SQL 语言编写。
- [not]deterministic(确定性):当函数返回不确定值时,该选项是为了防止"复制"时的不一致性。如果函数总是对同样的输入参数产生同样的结果,则被认为是"确定的",否则就是"不确定"的。例如函数返回系统当前的时间,返回值是不确定的。如

果既没有给定 deterministic 也没有给定 not deterministic，默认就是 not deterministic。

- {contains sql|no sql|reads sql data|modifies sql data}：指明子程序使用 SQL 语句的限制。contains sql 表示函数体中不包含读或写数据的语句（例如 set 命令等）。no sql 表示函数体中不包含 SQL 语句。reads sql data 表示函数体中包含 select 查询语句，但不包含更新语句。modifies sql data 表示函数体包含更新语句。如果上述选项没有明确指定，默认是 contains sql。
- sql security{definer|invoker}：设置执行权限。sql security 用于指定函数的执行许可。definer 表示该函数只能由创建者调用。invoker 表示该函数可以被其他数据库用户调用，默认值是 definer。
- comment 'string'：函数添加功能说明等注释信息。

（6）begin...end：函数体起止符。内含由 function_body_statements 描述的函数要实现的任务，一般由 begin...end 来描述任务代码的起止。函数体内要有“return return_values；”语句，表示函数返回值表达式。

2. 创建自定义函数举例

【例 7-6】 创建一个函数，计算长方形的面积。

代码和运行结果如下：

```
mysql > delimiter //
mysql > create function rectangle_area(long1 int,wide1 int) returns int
    -> begin
    -> return long1 * wide1;
    -> end //
Query OK, 0 rows affected (0.06 sec)
mysql > delimiter ;
```

【例 7-7】 创建一个名为 func_course 的函数返回表 course 中的指定课程号的课程名。

代码和运行结果如下：

```
mysql > delimiter &&
mysql > create function func_course(c_no varchar(6))
    -> returns char(6)
    -> begin
    ->     return (select cname from course
    ->              where courseno = c_no);
    -> end &&
       Query OK, 0 rows affected (0.00 sec)
mysql > delimiter ;
```

上述代码中，该函数的参数为 c_no，返回值是 char 类型。select 语句从 course 表查询 courseno 值等于 c_no 的记录，并将该记录的 cname 字段的值返回。执行结果显示，存储函数已经创建成功。该函数的使用和 MySQL 内部函数的使用方法一样，可以通过 select 语句调用函数。

3. 调用自定义函数

在 MySQL 系统中，因为函数和数据库相关，如果要调用函数，需要打开相应的数据库或指定数据库名称。存储函数的调用与 MySQL 内部函数的调用方式相同。

【例 7-8】 分别调用函数 rectangle_area()和 func_course。

代码和运行结果如下：

```
mysql > select rectangle_area(5,4);
        +-----------+
        | f_area(5,4)  |
        +-----------+
        |        20   |
        +-----------+
        1 row in set (0.00 sec)
mysql > select func_course('c08123');
        +------------------+
        | func_course('c08123') |
        +------------------+
        | 金融学              |
        +------------------+
        1 row in set (0.00 sec)
```

7.2.2 函数的维护管理

函数的维护包括查看函数的定义、修改函数的定义以及删除函数的定义等内容。

1. 查看函数的定义

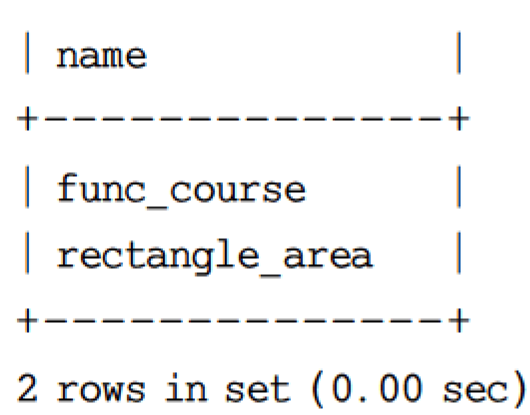
(1) 查看当前数据库中所有的自定义函数信息。例如：

```
show function status;                    //函数较少时用
show function status like 模式;          //自定义函数较多时用
```

(2) 查看指定数据库(如 teaching)中的所有自定义函数名。例如，可使用 SQL 语句。

代码和运行结果如下：

```
mysql > select name from MySQL.proc
      -> where db =  'teaching' and type = 'function';
```



```
          +--------------+
          | name          |
          +--------------+
          | func_course   |
          | rectangle_area  |
          +--------------+
          2 rows in set (0.00 sec)
```

(3) 可以查看指定函数名的详细信息。使用 MySQL 命令：

```
mysql > show create function func_name;
```

(4) 函数的信息都保存在 information_schema 数据库中的 routines 表中，可以使用 select 语句检索 routines 表，查询函数的相关信息。例如，查看函数 func_course 信息：

```
mysql > select * from information_schema.routines
    -> where routine_name = 'func_course';
```

2. 函数定义的修改

由于函数保存的仅仅是函数体，而函数体实际上是一组 MySQL 表达式，因此函数自身不保存任何用户数据。当函数的函数体需要更改时，可以使用 drop function 语句暂时将函数的定义删除，然后使用 create function 语句重新创建相同名字的函数即可。这种方法对于以后要介绍的存储过程、视图、触发器等数据库对象的修改同样适用。

MySQL 中修改函数的语句的语法形式如下：

```
alter function sp_name [characteristic … ];
```

【例 7-9】 修改存储函数 func_course 的定义。将读写权限改为 reads sql data，并加上注释信息“find function name”。

代码和运行结果如下：

```
mysql > alter function func_course
    -> reads sql data
    -> comment 'find function name';
       Query OK, 0 rows affected (0.00 sec)
mysql > select SPECIFIC_NAME,SQL_DATA_ACCESS,
    -> routine_comment from information_schema.Routines
    -> where routine_name = 'func_course';
       +------------+------------+-------------+
       | SPECIFIC_NAME | SQL_DATA_ACCESS | routine_comment  |
       +------------+------------+-------------+
       | func_course   | READS SQL DATA  | find function name|
       +------------+------------+-------------+
       1 row in set (0.01 sec)
```

3. 函数定义的删除

使用 MySQL 命令“drop function func_name”删除自定义函数。例如，删除 get_name()函数命令如下：

```
mysql > drop function get_name;
```

另外，函数的创建和管理还可以利用 MySQL Workbench 工具实现。

启动 MySQL Workbench 工具，单击实例 mysql57。在导航区 Navigator 下的 Schemas 区域，选择当前数据库 teaching。在 teaching 数据库中展开 functions 选项，可以查看到前面创建的函数。也可以选择一个函数，如 func_course，在如图 7-2 所示的弹出菜单中执行 Create Function、Alter Function 或 Drop Function 命令，实现对函数的创建、修改和删除等管理操作。

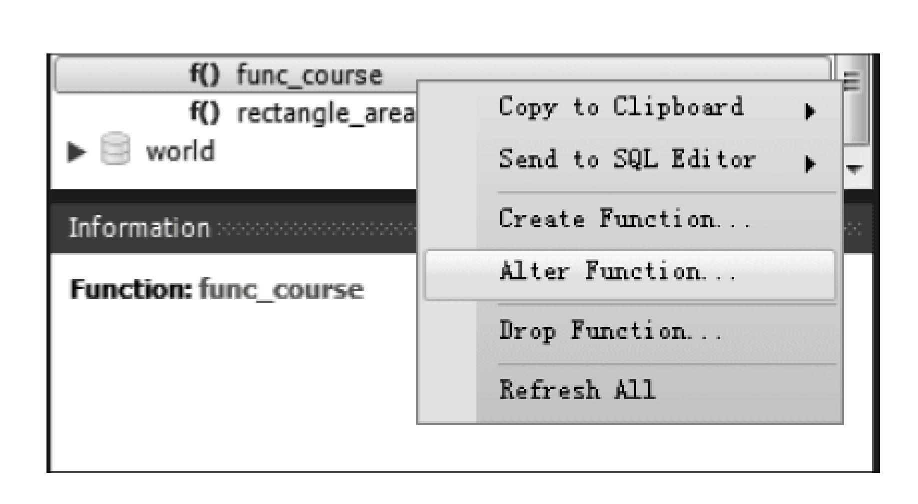


图 7-2 函数的管理

7.3 MySQL 的控制流语句

使用 MySQL 语言编程，可以通过控制流语句实现程序的顺序、选择和循环三种基本结构，乃至编写出能够解决较为复杂问题的存储过程、函数和触发器等。下面介绍相关的控制

流语句的使用方法。

7.3.1 条件控制语句

1. if 语句

if 语句用来进行条件判断，根据不同的条件执行不同的操作。该语句在执行时首先判断 if 后的条件是否为真，为真则执行 then 后的语句，如果为假则继续判断 if 语句直到为真为止，当以上都不满足时则执行 else 语句后的内容。if 语句的表示形式如下：



if 语句

```
if condition then
        …
[else condition then]
  …
[else]
…
endif
```

【例 7-10】 创建函数 exam_if，通过 if…then…else 结构首先判断传入参数的值是否为 10，如果是则输出 1，如果不是则再判断该传入参数的值是否为 20，如果是则输出 2，当以上条件都不满足时输出 3。然后调用函数 exam_if。

代码和运行结果如下：

```
mysql> delimiter //
mysql> create function exam_if(x int)
   -> returns int
   -> wosql
   -> begin
   -> if x = 10 then set x = 1;
   -> elseif x = 20 then set x = 2;
   -> else set x = 3;
   -> end if;
   -> return x;
   -> end //
      Query OK, 0 rows affected (0.00 sec)
mysql> delimiter ;
mysql> select exam_if(77);
        +-----------+
        | exam_if(77) |
        +-----------+
        |          3 |
        +-----------+
        1 row in set (0.00 sec)
```

2. case 语句

case 语句为多分支语句结构，该语句首先从 when 后的 value 中查找与 case 后的 value 相等的值，如果查找到则执行该分支的内容，否则执行 else 后的内容。



case 语句

case 语句表示形式如下：

```
case value
      when value then …
      [when valuethen…]
      [else…]
end case
```

其中,value 参数表示条件判断的变量;when...then 中的 value 参数表示变量的取值。

case 语句还有另一种语法表示结构:

```
case
     when value then …
     [when valuethen…]
     [else…]
end case
```

【例 7-11】 创建函数 exam_case,通过 case 语句首先判断传入参数的值是否为 10,如果条件成立则输出 1,如果条件不成立则再判断该传入参数的值是否为 20,如果成立则输出 2,当以上条件都不满足时输出 3。

代码和运行结果如下:

```
mysql> delimiter //
mysql> create function exam_case( x int)
    -> returns int
    -> nosql
    -> begin
    -> case x
    -> when 10 then set x = 1;
    -> when 20 then set x = 2;
    -> else set x = 3;
    -> end case;
    -> return x;
    -> end //
        Query OK, 0 rows affected (0.03 sec)
mysql> delimiter ;
mysql> select exam_case(17);
         +------------+
         | exam_case(17) |
         +------------+
         |          3  |
         +------------+
         1 row in set (0.02 sec)
```

3. 条件判断函数

MySQL 中常用的条件控制函数有 if()、ifnull()以及 case 函数,这些函数的功能是根据条件表达式的值返回不同的值,而且函数可以在 MySQL 客户机中直接调用。条件判断函数用来在 SQL 语句中进行条件判断。根据是否满足判断条件,SQL 语句执行不同的分支。

(1) if()函数。if(condition,v1,v2)函数中 condition 为条件表达式,当 condition 的值为 true 时,函数返回 v1 的值,否则返回 v2 的值。

【例 7-12】 从表 student 中查询学号 studentno,入学成绩 entrance。成绩大于等于 800 分,显示“pass!”。否则,显示“bye.”,输出前 5 条记录。

代码和运行结果如下：

```
mysql > select studentno, entrance, if(entrance > = 800, 'pass', 'bye! ')
    -> from student limit 5;
        +------------+---------+------------------------------+
        | studentno   | entrance | if(entrance > = 800, 'pass! ', 'bye. ') |
        +------------+---------+------------------------------+
        | 18122210009 |      789 | bye.                         |
        | 18122221324 |      879 | pass!                        |
        | 18125111109 |      789 | bye.                         |
        | 18125121107 |      777 | bye.                         |
        | 18135222201 |      867 | pass!                        |
        +------------+---------+------------------------------+
        5 rows in set (0.00 sec)
```

(2) ifnull()函数。ifnull(v1,v2)函数中，如果 v1 的值为 null，则该函数返回 v2 的值；如果 v1 的值不为 null，则该函数返回 v1 的值。

(3) case 函数。case 函数有两种格式，验算过程相近。如果表达式的值等于 when 语句中某个"值 n"，则 case 函数返回值为"结果 n"；如果与所有的"值 n"都不相等，case 函数返回值为"其他值"。

```
case 表达式 1                    //格式 1
when 值 1 then 结果 1
[when 值 2 then 结果 2 ]…
[else 其他值 ]
end
case                            //格式 2
when 表达式 1 then 值 1
[when 表达式 2 then 值 2… ]
[else 其他值]
end
```

可以参照例 7-12 中 if()函数的用法，理解 ifnull()函数和 case 函数的用法。

7.3.2 循环语句

1. while 循环语句



while 语句

while 循环语句执行时首先判断 condition 条件是否为真，如果是则执行循环体，否则退出循环。该语句表示形式如下：

```
while condition do
…
end while;
```

【例 7-13】 定义函数 exam_while，应用 while 语句求 1 到 100 项的和。

分析：首先定义变量 m 和 sum，分别用来控制循环的次数和保存前 100 项的和，当变量 m 的值小于或等于 100 时，使 sum 的值加 m，并同时使 m 的值增 1。直到 m 大于 100 时退出循环并输出结果。

代码和运行结果如下：

```
mysql > delimiter //
```

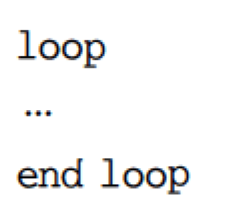
```
mysql > create function exam_while(n int) returns int
    -> nosql
    -> begin
    -> declare sum int default 0;
    -> declare m int default 1;
    -> while m < = n do
    -> set sum = sum + m;
    -> set m = m + 1;
    -> end while;
    -> return sum;
    -> end //
       Query OK, 0 rows affected (0.01 sec)
mysql > delimiter ;
mysql > select exam_while(100);
        +---------------+
        | exam_while(100) |
        +---------------+
        |          5050 |
        +---------------+
        1 row in set (0.00 sec)
```

2. loop 循环语句



loop 语句

loop 循环语句是没有内置的循环条件，但可以通过 leave 语句退出循环。loop 语句表示形式如下：



```
loop
…
end loop
```

loop 允许某特定语句或语句群的重复执行，实现一个简单的循环构造，其中中间省略的部分是需要重复执行的语句。在循环内的语句一直重复直至循环被退出，退出循环应用 leave 语句。

leave 语句经常和 begin…end 语句或循环一起使用，其结构如下：

```
leave label
```

label 是语句中标注的名字，这个名字是自定义的。加上 leave 关键字就可以用来退出被标注的循环语句。

【例 7-14】 定义函数 exam_loop，应用 loop 语句求 1～100 之和。通过 leave 语句退出循环并输出结果。

代码和运行结果如下：

```
mysql > delimiter//
mysql > create function exam_loop(n int) returns int
    -> nosql
    -> begin
    -> declare sum int default 0;
    -> declare k int default 1;
    -> loop_label:loop
    -> set sum = sum + k;
    -> set k = k + 1;
```

```
    -> if k > n then
    -> leave loop_label;
    -> end if;
    -> end loop;
    -> return sum;
    -> end //
       Query OK, 0 rows affected (0.00 sec)
mysql > delimiter;
mysql > select exam_loop(100);
        +--------------+
        | exam_loop(100) |
        +--------------+
        |        5050    |
        +--------------+
        1 row in set (0.00 sec)
```

循环语句中还有一个 iterate 语句，它可以出现在 loop、repeat 和 while 语句内，意为结束本次循环。该语句格式如下：

```
iterate label
```

该语句的格式与 leave 大同小异，区别在于：leave 语句是结束循环，而 iterate 语句是结束本次循环。

【例 7-15】 定义函数 exam_iterate，应用 while 语句和 iterate 语句求 1～100 的偶数之和。通过 leave 语句退出循环并输出结果。



iterate 语句的用法

代码和运行结果如下：

```
mysql > delimiter //
mysql > create function exam_iterate(n int) returns int
    -> nosql
    -> begin
    -> declare sum char(20) default 0;
    -> declare s int default 0;
    -> add_num: while true do
    -> set s = s + 1;
    -> if (s % 2 = 0) then
    -> set sum = sum + s;
    -> else
    -> iterate add_num;
    -> end if;
    -> if (s = n) then
    -> leave add_num;
    -> end if;
    -> end while add_num;
    -> return sum;
    -> end;
    -> //
       Query OK, 0 rows affected (0.00 sec)
mysql > delimiter ;
mysql > select exam_iterate(100);
        +----------------+
        | exam_iterate(100) |
```

```
+----------------+
|          2550  |
+----------------+
1 row in set (0.00 sec)
```

3. repeat 循环语句



repeat 语句

repeat 循环语句是先执行一次循环体，之后判断 condition 条件是否为真，真则退出循环，否则继续执行循环。repeat 语句表示形式如下：

```
repeat
  …
until condition
end repeat
```

【例 7-16】 定义函数 exam_repeat，应用 repeat 语句求 1～50 的和。

代码和运行结果如下：

```
mysql> delimiter //
mysql> create function exam_repeat (n int) returns int
    -> nosql
    -> begin
    -> declare sum int default 0;
    -> declare p int default;
    -> repeat
    -> set sum = sum + p;
    -> set p = p + 1;
    -> until p > n
    -> end repeat;
    -> return sum;
    -> end //
       Query OK, 0 rows affected (0.00 sec)
mysql> delimiter ;
mysql> select exam_repeat(50);
       +----------------+
       | exam_repeat(50)  |
       +----------------+
       |        1275    |
       +----------------+
       1 row in set (0.00 sec)
```

在编写循环类的程序时，应特别注意函数参数和函数体内的变量值的变化情况，稍有不慎，就会出现问题，诸如数据错误、死循环等。

7.4 小　　结

利用 MySQL 语言编程，需要先掌握常量、变量、函数、表达式、关键词等语言因素的使用方法，并在此基础上利用创建自定义函数应用这些语言因素去表达或描述算法，实现编写程序进行一系列的操作。较为复杂的算法需要利用选择和循环语句去实现，在 MySQL 中，这些控制流语句必须在存储过程或存储函数中才能够使用。存储函数都是用户自己定义的

SQL 语句的集合，存储在服务器端，只要调用就可以在服务器端执行。

在本章学习过程中，需要重点掌握如下知识点：

- 变量的分类、定义和使用方法。尤其是学会会话变量的使用方法。
- 存储函数的定义和使用方法。
- 预处理语句的定义和执行方法。
- 控制流语句的语句格式和使用方法。特别是选择结构和循环结构的各类语句，是 MySQL 语句的主要组成部分，是 MySQL 实现较为复杂算法的基础。
- 选择结构分为 if 语句和 case 语句，适合判断类的算法处理。
- 循环语句包括 while、repeat、loop 等方式，适合处理重复执行的语句块。

习　题　7

1. 选择题

(1) 在 MySQL 语句中，可以匹配 0 个到多个字符的通配符是________。

A. *　　B. %　　C. ?　　D. -

(2) MySQL 提供的单行注释语句可以是使用________开始的一行内容。

A. /*　　B. #　　C. {　　D. /

(3) 在 MySQL 中会话变量前面的字符为________。

A. *　　B. #　　C. @@　　D. @

(4) 若要计算表中数据的平均值，可以使用的函数是________。

A. sqrt　　B. avg　　C. square　　D. count

(5) ________语句用于执行预处理语句。

A. prepare　　B. deallocate　　C. execute　　D. using

2. 思考题

(1) MySQL 的语言要素有哪些？主要作用是什么？

(2) 如何定义变量，如何给变量赋值？

(3) 流程控制语句包括哪些类型？各自的作用是什么？

(4) MySQL 语句共分几类，各自的主要功能是什么？

3. 上机练习题

(1) 利用预处理 SQL 语句输出数据库 teaching 中 teacher 表的前 5 行记录的部分数据。

(2) 创建函数 casetwo，通过 case 语句首先判断传入参数的值是否为 7，如果条件成立则输出 1，如果条件不成立则再判断该传入参数的值是否为 14，如果成立则输出 2，当以上条件都不满足时输出−1。

(3) 定义函数 exthree，应用 while 语句编程求 50 到指定整数的所有奇数之和。

(4) 定义函数 exfour，应用 while 语句和 iterate 语句求 100～150 的偶数之和。通过 leave 语句退出循环并输出结果。

第 8 章　存储过程、游标和触发器

存储过程(Stored Procedure)是一组完成特定功能的 MySQL 语句的集合,即将一些固定的操作集中起来由 MySQL 服务器来完成,应用程序只需调用它就可以实现某个特定的任务。存储过程是可以通过用户、其他存储过程或触发器来调用执行的。

在 MySQL 中的游标(Cursor)是一种实现对 select 结果集中的数据进行访问和处理的机制,允许用户访问单独的数据行。MySQL 中的游标一般通过存储过程来实现其操作。

触发器(Trigger)是一种特殊的存储过程。触发器通常在特定的表上定义,当该表的相应事件发生时自动执行,用于实现强制业务规则和数据完整性等。

事件(Event)又称事件调度器(Event Scheduler),有时也可称为临时触发器(Temporal Triggers),因为事件调度器是基于特定时刻或时间周期触发来执行某些任务,而触发器是基于对表进行操作所产生的事件触发的。

本章将介绍存储过程、游标、触发器和事件的基本概念,以及它们的创建和管理的基本操作。

8.1 存 储 过 程

本节从认识存储过程着手,学习创建、执行、修改和删除存储过程的方法。包括如何创建基本的存储过程、带有输入输出参数的存储过程、带有流程控制语句的复杂存储过程。还要介绍一下利用 MySQL Workbench 管理存储过程的基本操作。

8.1.1 认识存储过程

在 MySQL 数据库中,利用存储过程可以保证数据的完整性,提高执行重复任务的性能和数据的一致性。例如,银行经常核算用户的利息,不同类别的用户的存款利率是不一样的。这就可以将计算利率的 SQL 代码写成一个存储过程。只要将客户的某一笔存款的存款时间和存款额数输入,调用这个存储过程就可以核算出用户的利息。存储过程在被调用的过程中,参数可以被传递和返回,出错代码也可以被检验。

存储过程主要应用于控制访问权限、为数据库表中的活动创建审计追踪、将关系到数据库及其所有相关应用程序的数据定义语句和数据操作语句分隔开。

1. 存储过程的优势

利用存储过程可以让系统达到如下目的:

(1) 提高了处理复杂任务的能力。主要用于在数据库中执行操作的编程语句,通过接收输入参数并以输出参数的格式向调用过程或批处理返回多个值。

（2）增强了代码的复用率和共享性。存储过程一旦创建即可在程序中调用任意多次，这可以改进应用程序的可维护性，并允许应用程序统一访问数据库。

（3）减少了网络中数据的流量。因为存储过程存储在服务器上，并在服务器上运行。一个需要数百行 MySQL 代码的操作可以通过一条执行过程代码的语句来执行，而不需要在网络中发送数百行代码。

（4）存储过程在服务器注册，加快了过程的运行速度。存储程序只在创建时进行编译，以后每次执行存储过程都不需要再重新编译，而一般 MySQL 语句每执行一次就编译一次，所以使用存储过程可提高数据库的执行速度。

（5）加强了系统的安全性。存储过程具有安全特性（例如权限）和所有权链接，用户可以被授予权限来执行存储过程而不必直接对存储过程中引用的对象具有权限。可以强制应用程序的安全性，参数化存储过程有助于保护应用程序不受 SQL 注入式攻击。

2. 创建存储过程格式

创建存储过程可以使用 create procedure 语句。要创建存储过程，必须具有 create routine 的权限。

create procedure 的语法格式如下：

```
create procedure sp_name([proc_parameter[,...]])
[characteristic ...] routine_body
```

说明：

- create procedure：创建存储过程的关键字。
- sp_name：存储过程的名称。需要在特定数据库中创建存储过程时，则要在名称前面加上数据库的名称，格式为 db_name. sp_name。
- proc_parameter：存储过程的参数列表。在使用参数时要标明参数名和参数的类型，当有多个参数的时候中间用逗号隔开。MySQL 存储过程支持三种类型的参数：输入参数、输出参数和输入输出参数，关键字分别是 in、out 和 inout。存储过程也可以不加参数，但是名称后面的括号是不可省略的。
- characteristic：存储过程的某些特征设定。参看函数定义说明。
- routine_body：存储过程体。包含在过程调用的执行语句中，这个部分总是以 begin 开始，以 end 结束。当然，当存储过程体中只有一个 SQL 语句时可以省略 begin…end 标志。

3. 调用存储过程

要使用这些已经定义好的存储过程就必须要通过调用的方式来实现。存储过程是通过 call 语句来调用的。执行存储过程需要拥有 execute 权限。execute 权限的信息存储在 information_schema 数据库下面的 user_privileges 表中。

在 MySQL 数据库系统中，存储过程是数据库对象，如果要执行其他数据库中的存储过程，需要打开相应的数据库或指定数据库名称。

MySQL 可以利用 call 命令调用存储过程，其语法格式如下：

```
call [dbname.]sp_name([parameter[,…]]);
```

说明：

(1) sp_name 为存储过程的名称，如果要调用某个特定数据库的存储过程，则需要在前面加上该数据库的名称。

(2) parameter 为调用该存储过程使用的参数，这条语句中的参数个数必须总是等于存储过程的参数个数。

8.1.2 存储过程的创建和管理



创建存储过程

1. 存储过程的创建和执行

【例 8-1】 创建存储过程 proc_stu，从数据库 teaching 的 student 表中检索出所有电话以 131 开头的学生的学号、姓名、出生日期和电话等信息。

代码和运行结果如下：

```
mysql > use teaching;
Database changed
mysql > delimiter //
mysql > create procedure proc_stu()
    -> reads sql data
    -> begin
    -> select studentno, sname, birthdate , phone
    -> from student
    -> where phone like '%131%' order by studentno ;
    -> end//
        Query OK, 0 rows affected (0.14 sec)
mysql > delimiter ;
```

调用存储过程 proc_stu()的代码和执行结果如下：

```
mysql > call proc_stu();
      +-----------+-------+------------+-----------+
      | studentno   | sname | birthdate    | phone       |
      +-----------+-------+------------+-----------+
      | 18122221324 | 何白露 | 2000 - 12 - 04 | 13178978999 |
      | 19123567897 | 赵既白 | 2002 - 08 - 04 | 13175689345 |
      +-----------+-------+------------+-----------+
      2 rows in set (0.05 sec)
      Query OK, 0 rows affected (0.08 sec)
```

【例 8-2】 创建存储过程 avg_score，输入课程号后，统计该课程的平均成绩。

代码和运行结果如下：

```
mysql > delimiter //
mysql > create procedure avg_score(in c_no char(6))
    -> begin
    -> select courseno, avg(final)
    -> from score
    -> where courseno = c_no ;
    -> end //
      Query OK, 0 rows affected (0.05 sec)
mysql > delimiter ;
```

调用存储过程 avg_score ()的代码和执行结果如下：

```
mysql> call avg_score('c05109');
          +--------+---------+
          | courseno | avg(final) |
          +--------+---------+
          | c05109   | 79.85714  |
          +--------+---------+
         1 row in set (0.06 sec)
         Query OK, 0 rows affected (0.06 sec)
```

【例 8-3】 创建存储过程 select_score()，用指定的学号和课程号为参数查询学生成绩。

分析：创建带多个输入参数的存储过程。

代码和运行结果如下：

```
mysql> delimiter $$
mysql> create procedure select_score(in s_no char(11),c_no char(6))
     -> begin
     -> select * from score
     -> where studentno = s_no and courseno = c_no ;
     -> end $$
         Query OK, 0 rows affected (0.01 sec)
mysql> delimiter ;
```

调用存储过程 select_score()的代码和执行结果如下：

```
mysql> call select_score('18125121107','c05109');
          +-----------+--------+-----+-----+
          | studentno    | courseno | daily | final |
          +-----------+--------+-----+-----+
          | 18125121107 | c05109   | 89.0  | 59.0  |
          +-----------+--------+-----+-----+
         1 row in set (0.00 sec)
         Query OK, 0 rows affected (0.02 sec)
```

【例 8-4】 创建存储过程 stu_score，统计指定同学的考试门数。

分析：在本存储过程中，输入参数为学号 s_no，输出参数为 count_num，select 语句用 count(*)计算指定学生的考试门数，最后将计算结果存入 count_num 中。调用有输出参数的存储过程时，可以通过会话变量 @c_num 实现。



带输出参数的存储过程

代码和运行结果如下：

```
mysql> delimiter //
mysql> create procedure stu_scores(in s_no char(11), out count_num int)
     -> reads SQL data
     -> begin
     -> select count( * ) into count_num from score
     -> where studentno = s_no;
     -> end //
         Query OK, 0 rows affected (0.00 sec)
mysql> delimiter ;
```

调用存储过程 stu_scores()的代码和执行结果如下：

```
mysql> call stu_scores('18125121107', @c_num );
          Query OK, 1 row affected (0.02 sec)
mysql> select @c_num;
           +--------+
           | @c_num |
           +--------+
           |      2 |
           +--------+
          1 row in set (0.00 sec)
```

说明：

(1) 存储过程是已保存的 MySQL 语句集合。对于一般的 select 语句，如果查询的数据要来自于多个表，可以使用多表连接或子查询等方式。

(2) 当调用存储过程时，MySQL 会根据提供的参数值，执行存储过程体中的 SQL 语句。

【例 8-5】 创建存储过程 do_query，输入指定学号，查看该生的成绩高于 85 分的科目数，如果超过两科，则输出 very good!，并输出该生的成绩单，否则输出 come on!。

分析：存储过程 do_query 中，利用 if 语句实现较为复杂的功能。该存储过程用 declare 语句声明了局部变量 AA。根据指定学号，统计该生高于 85 分的科目数，并使用 select into 语句为变量 AA 赋值，然后根据 AA 的值进行判断。

代码和运行结果如下：

```
mysql> delimiter //
mysql> create procedure do_query(in s_no char(11), out str char(12))
     -> begin
     -> declare AA tinyint default 0;
     -> select count( * ) into AA from score
     -> where studentno = s_no and final > 85;
     -> if AA >= 2 then
     ->   begin
     ->      set str = 'very good! ';
     ->      select * from score where studentno = s_no;
     ->    end;
     -> elseif  AA < 2 then
     -> set str = 'come on! ';
     -> end if;
     -> end //
          Query OK, 0 rows affected (0.02 sec)
mysql> delimiter ;
```

调用存储过程 do_query()的代码和执行结果如下：

```
mysql> call do_query('18125111109',@str);
           +-------------+----------+-------+-------+
           | studentno   | courseno | daily | final |
           +-------------+----------+-------+-------+
           | 18125111109 | c08106   |  79.0 | 94.0  |
```

```
        | 18125111109 | c08123   |   85.0 | 87.0  |
        | 18125111109 | c08171   |   77.0 | 87.0  |
        +-----------+--------+-----+-----+
        3 rows in set (0.00 sec)
        Query OK, 0 rows affected (0.02 sec)
mysql> select @str;
        +----------+
        | @str      |
        +----------+
        | very good! |
        +----------+
        1 row in set (0.00 sec)
mysql> call do_query('18122210009',@str);
        Query OK, 1 row affected (0.00 sec)
mysql> select @str;
        +--------+
        | @str    |
        +--------+
        | come on! |
        +--------+
        1 row in set (0.00 sec)
```

【例 8-6】 创建一个存储过程,向 score 表中插入一行记录,然后创建另一存储过程 do_outer(),调用存储过程 do_insert(),并查询输出 score 表中插入的记录。



调用存储过程

分析:利用存储过程调用其他存储过程。在调用存储过程 do_outer()时,先执行第一个存储过程 do_insert(),插入了一行记录,然后再执行后面的语句,输出查询结果。

代码和运行结果如下:

```
 -- 先创建第 1 个存储过程 do_insert()
mysql> create procedure do_insert()
    -> insert into score values('18125111109', 'c05109',89,92);
        Query OK, 0 rows affected (0.03 sec)
 -- 创建第 2 个存储过程 do_outer(),调用 do_insert()
mysql> delimiter $$
mysql> create procedure do_outer()
    -> begin
    -> call do_insert();
    -> select * from score
    -> where studentno = '18125111109';
    -> end $$
        Query OK, 0 rows affected (0.02 sec)
```

调用存储过程 do_outer()的代码和执行结果如下:

```
mysql> call do_outer();
        +-----------+--------+-----+-----+
        | studentno   | courseno | daily | final |
        +-----------+--------+-----+-----+
        | 18125111109 | c05109   | 89.0  | 92.0 |
        | 18125111109 | c08106   | 79.0  | 94.0 |
```

```
| 18125111109 | c08123   | 85.0  | 87.0 |
| 18125111109 | c08171   | 77.0  | 87.0 |
+-----------+--------+-----+-----+
4 rows in set (0.30 sec)
Query OK, 0 rows affected (0.32 sec)
```

2. 查看存储过程的定义

存储过程和函数创建以后,用户可以查看存储过程和函数的状态及定义。用户可以通过 show status 语句来查看存储过程和函数的状态,也可以通过 show create 语句来查看存储过程和函数的定义。用户也可以通过查询 information_schema 数据库下的 Routines 表来查看存储过程和函数的信息。在前面学习存储函数的基础上,下面给出已经验证过的查看存储过程的状态和定义的方法的例子:

```
mysql> show procedure status like 'do_ % ';
mysql> show create procedure do_outer;
mysql> select * from information_schema.routines
    -> where routine_name = 'do_outer ' ;
mysql> show create procedure do_outer;
```

3. 条件和处理程序的定义

默认情况下,MySQL 存储程序运行过程中发生错误时,将自动终止程序的执行。此时,数据库开发人员有时希望自己控制程序的运行流程,并不希望 MySQL 自动终止存储程序的执行,MySQL 的错误处理机制可以帮助数据库开发人员自行控制程序流程。

定义条件和处理程序是事先定义程序执行过程中可能遇到的问题,并且可以在处理程序中定义解决这些问题的办法。这种方式可以提前预测可能出现的问题,并提出解决办法。这样可以增强程序处理问题的能力,避免程序异常停止。MySQL 中都是通过 declare 关键字来定义条件和处理程序的。通过定义条件来对可能涉及的错误以及子程序中的一般流程进行控制。

(1) 定义条件。在 MySQL 中定义条件的基本语法如下:

```
declare condition_name condition for condition_type;
condition_type:
sqlstate [value] sqlstate_value|mysql_error_code
```

说明:

① condition_name:表示错误触发条件的名称。

② condition_type:表示条件的类型,分为 MySQL 错误代码或者 ANSI 标准错误代码。sqlstate_value 是长度为 5 的字符串类型错误代码;mysql_error_code 为数值型错误代码。例如 ERROR 1147 (42S07)中,sqlstate_value 的值为字符串"42S07",mysql_error_code 为 1147。

③ 此语句指定需要特殊处理的条件,将指定的错误条件与一个名字联系起来。这个名字即错误名,可以在随后的定义处理程序中的 declare handler 语句中应用。

【例 8-7】 定义"ERROR 1147 (42S07)"这个错误,名称为 cannot_found。可以用两种不同的方法来定义,代码如下:

```
-- 使用 sqlstate_value 方法定义
declare cannot_found condition for sqlstate'42S07';
-- 使用 mysql_error_code 方法定义
declare cannot_found condition for 1147;
```

(2) 定义处理程序。MySQL 中定义处理程序的基本语法如下：

```
declare handler_type handler for condition_value[, … ] sp_statement
handler_type:
continue|exit
condition_value:
sqlstate [value]sqlstate_value|condition_name
|sqlwarning|not found|sqlexception|mysql_error_code
```

说明：

① handler_type：错误处理类型，取值包括 continue 和 exit。continue 表示遇到错误不处理，继续执行其他 MySQL 语句；exit 表示遇到错误马上退出其他 MySQL 语句的执行。

② condition_value：错误触发条件，表示满足什么条件时，自定义错误处理程序开始运行，错误触发条件定义了自定义错误处理程序运行的时机。具体包括如下取值：

- sqlstate [value] sqlstate_value ：长度为 5 的字符串类型错误代码。
- condition_name：表示 declare condition 定义的错误条件名称。
- sqlwarning：匹配所有以 01 开头的 sqlstate 错误代码。
- not found：匹配所有以 02 开头的 sqlstate 错误代码。
- sqlexception：匹配其他非 sqlwarning 和 not found 捕获的错误代码。
- mysql_error_code：为数值型错误代码。

③ sp_statement：自定义错误处理程序，即遇到定义的错误时，MySQL 会立即执行自定义错误处理程序中的 MySQL 语句，自定义错误处理程序也可以是一个 begin…end 语句块。

【例 8-8】 定义条件和处理程序示例。

代码和运行结果如下：

```
-- 首先建立测试表 mytest
mysql> create table mytest(tf1 int,primary key(tf1));
mysql> delimiter //
mysql> create procedure handlermytest()
    -> begin
    -> declare continue handler for sqlstate '23000' set @x2 = 1;
    -> set @x = 1;
    -> insert into mytest values(1);
    -> set @x = 2;
    -> insert into mytest values(1);
    -> set @x = 3;
    -> select @x,@x2;
    -> end;
    -> //
      Query OK, 0 rows affected (0.00 sec)
mysql> delimiter ;
```

调用存储过程 handlermytest()的代码和执行结果如下：

```
mysql > call handlermytest();
          +----+-----+
          | @x  | @x2 |
          +----+-----+
          |  3  |  1   |
          +----+-----+
          1 row in set (0.03 sec)
          Query OK, 0 rows affected (0.03 sec)
```

说明：

① 定义了异常处理程序后，此时 MySQL 遇到错误也会按照异常定义那样继续执行；但只有第 1 条数据被插入到表中，此时用户变量@x=3 说明已经执行到了结尾。

② 自定义错误触发条件以及自定义错误处理程序可以在触发器、函数以及存储过程中使用。实际软件开发过程中，建议数据库开发人员建立清晰的错误处理规范，必要时可以将自定义错误触发条件、自定义错误处理程序封装在一个存储程序中。

8.1.3 修改存储过程

有两种方法可以修改存储过程，一种方法是删除并重新创建存储过程，这种方法和创建存储过程一样。另一种方法是使用 alter procedure 语句进行修改。使用 alter procedure 语句修改存储过程的某些参数，修改存储过程的语法格式如下：

```
alter procedure sp_name [characteristic ...]
```

【例 8-9】 修改存储过程 do_insert()的定义。将读写权限改为 modifies sql data，并指明调用者可以执行。

代码和运行结果如下：

```
mysql > alter procedure do_insert
     -> modifies sql data
     -> sql security invoker;
        Query OK, 0 rows affected (0.02 sec)
```

8.1.4 删除存储过程

删除存储过程可以使用 drop procedure 语句实现。使用 drop procedure 删除已经存在的存储过程的语法格式如下：

```
drop procedure [if exists] sp_name
```

说明：

(1) sp_name 是要删除的存储过程的名称。

(2) if exists 子句是 MySQL 的扩展，如果程序或函数不存在，它防止发生错误。

例如，删除存储过程 do_update()的代码如下：

```
mysql > drop procedure if exists do_insert;
```

8.1.5 存储过程与函数的比较

1. 存储过程与函数之间的共同特点

(1) 存储过程或者函数可以重复使用,可以减少数据库开发人员,尤其是应用程序开发人员的工作量。

(2) 使用存储过程或者函数可以增强数据的安全访问控制。可以设定只有某些数据库用户才具有某些存储过程或者函数的执行权。

2. 存储过程与函数之间的不同之处

(1) 函数必须有且仅有一个返回值,且必须指定返回值为字符串、数值两个数据类型。存储过程可以没有返回值,也可以有返回值,甚至可以有多个返回值,所有的返回值需要使用 out 或 inout 参数定义。

(2) 函数体内可以使用 select…into 语句为某个变量赋值,但不能使用 select 语句返回结果集。存储过程则没有这方面的限制,存储过程甚至可以返回多个结果集。

(3) 函数可以直接嵌入到 SQL 语句或 MySQL 表达式中,最重要的是函数可以用于扩展标准的 SQL 语句。存储过程一般需要单独调用,并不会嵌入到 SQL 语句中使用,调用时需要使用 call 关键字。

(4) 函数中的函数体限制比较多,例如函数体内不能使用以显式或隐式方式打开、开始或结束事务的语句,如 start transaction、commit、rollback 或 set autocommit=0 等语句;不能在函数体内使用预处理 SQL 语句。存储过程的限制相对就比较少,基本上所有的 SQL 语句或 MySQL 命令都可以在存储过程中使用。

(5) Java、PHP 等应用程序调用函数时,通常将函数封装到 SQL 字符串中进行调用;而调用存储过程时,必须使用 call 关键字进行调用,如果应用程序希望获取存储过程的返回值,应用程序必须给存储过程的 out 参数或 inout 参数传递 MySQL 会话变量,才能通过该会话变量获取存储过程的返回值。

利用 Workbench 管理存储过程

8.1.6 利用 MySQL Workbench 工具管理存储过程

利用 MySQL Workbench 工具管理存储过程主要包括对存储过程的创建、修改、查看、删除和执行操作。

1. 查看存储过程

(1) 启动 MySQL Workbench 工具,单击实例 mysql57,选择当前数据库 teaching。

(2) 在 teaching 数据库中选择 Stored Procedures 选项,展开 Stored Procedures 文件夹,可以看到已经创建的存储过程,如图 8-1 所示。在弹出菜单中,可以实现对存储过程进行复制到剪贴板、发送到 SQL 编辑器、新建、修改、删除和刷新等一系列操作。

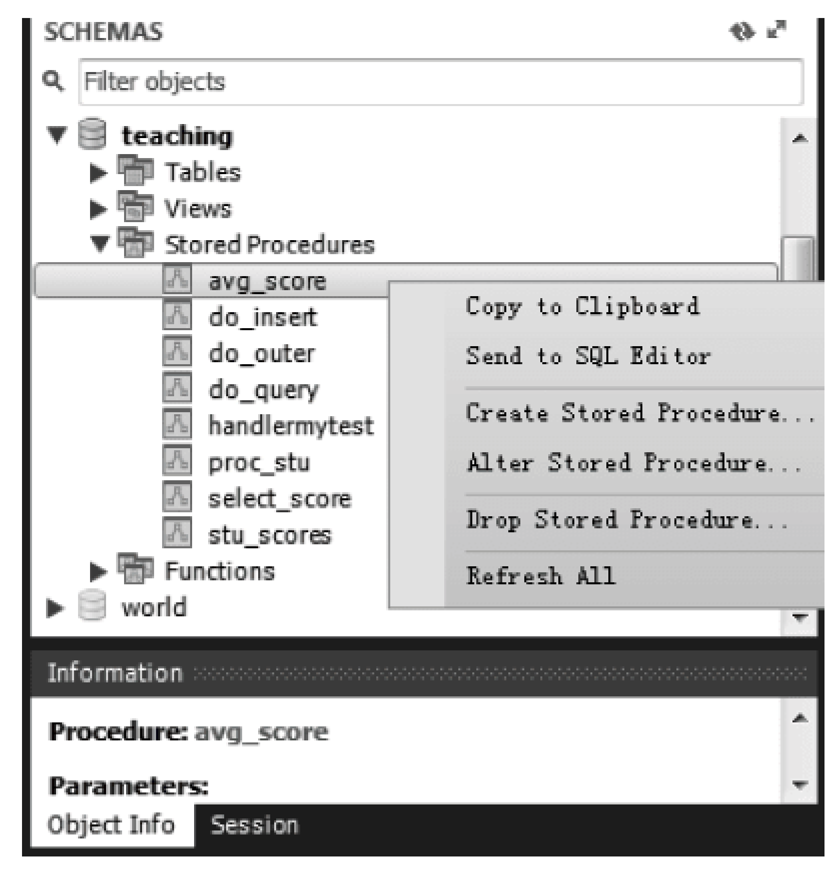


图 8-1 查看存储过程

2. 定义存储过程

(1) 创建存储过程,选择执行 Create Stored Procedures 命令。则会进入如图 8-2 所示的创建存

储过程的初始界面 new_procedure-Routine。

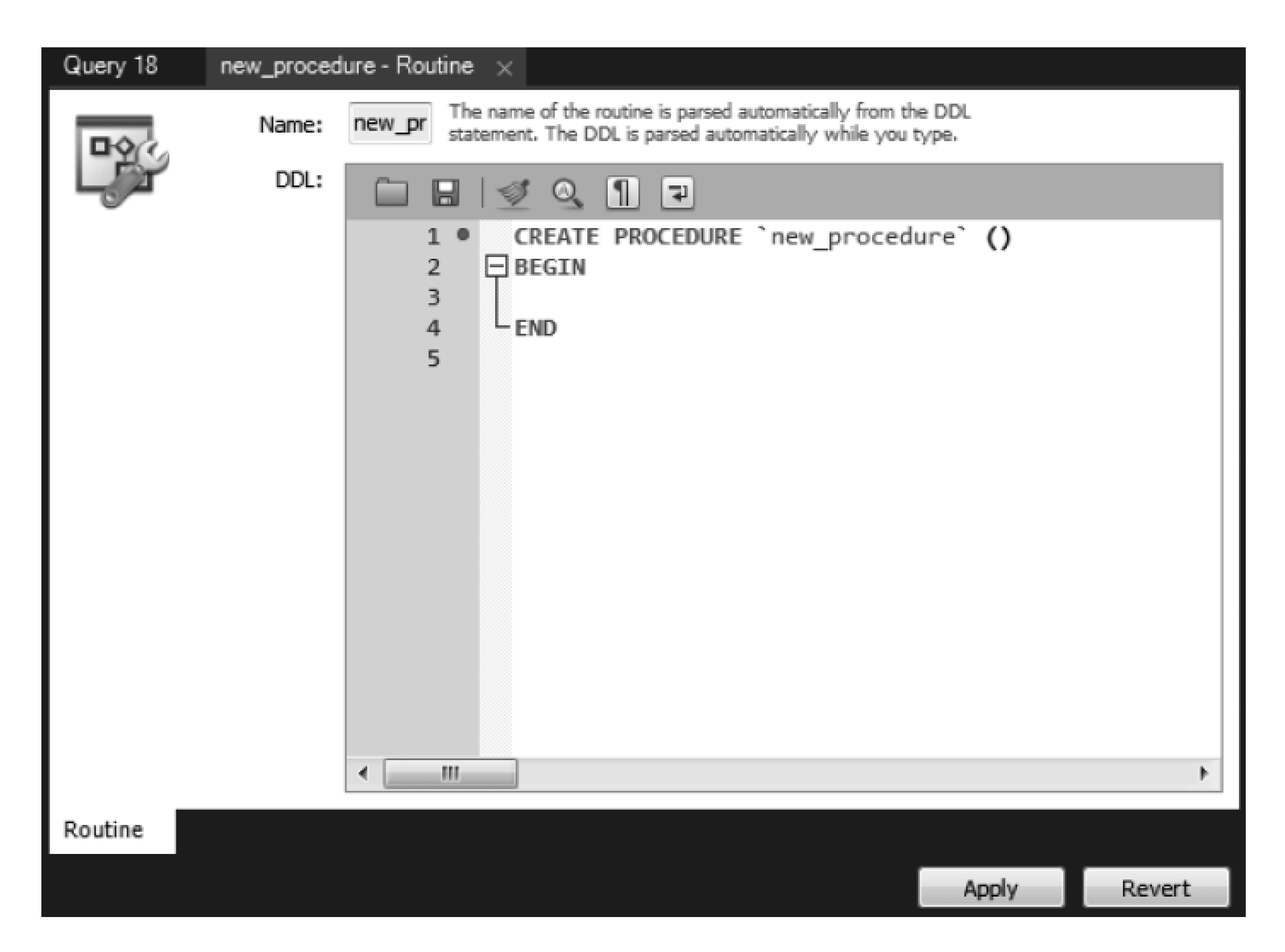


图 8-2　创建存储过程的初始界面

(2) 在编辑界面输入存储过程名 do_case,在 begin…end 之间输入存储过程程序体的代码,如图 8-3 所示。若有错误,则会在行前提示。

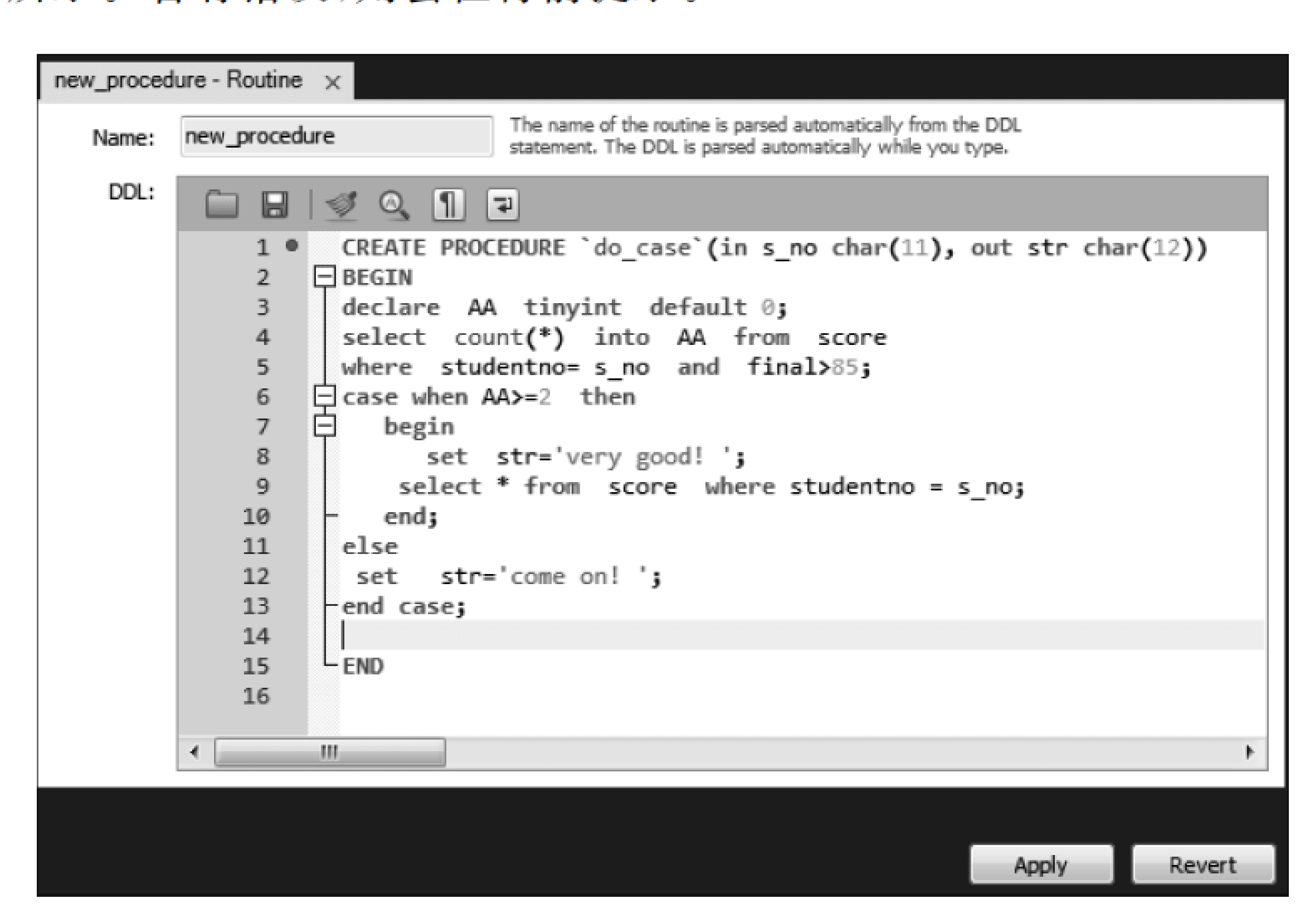


图 8-3　输入存储过程的名称和代码

(3) 检查无误后,单击 Apply 按钮,进入如图 8-4 所示的代码对话框中,这是要向数据库 teaching 中存储的脚本。脚本内容为例 8-5 的 do_query 存储过程,将 if 语句改为 case 语句模式来实现。

(4) 单击 Apply 按钮,进入如图 8-5 所示的对话框中,可以通过 Show logs(Hide logs)转换按钮查看存储过程的信息记录(Massage Log)窗口中的信息。可以查看到成功创建存储过程的提示:SQL script was successfully applied to the database。单击 Finish 按钮完成存储过程的创建。

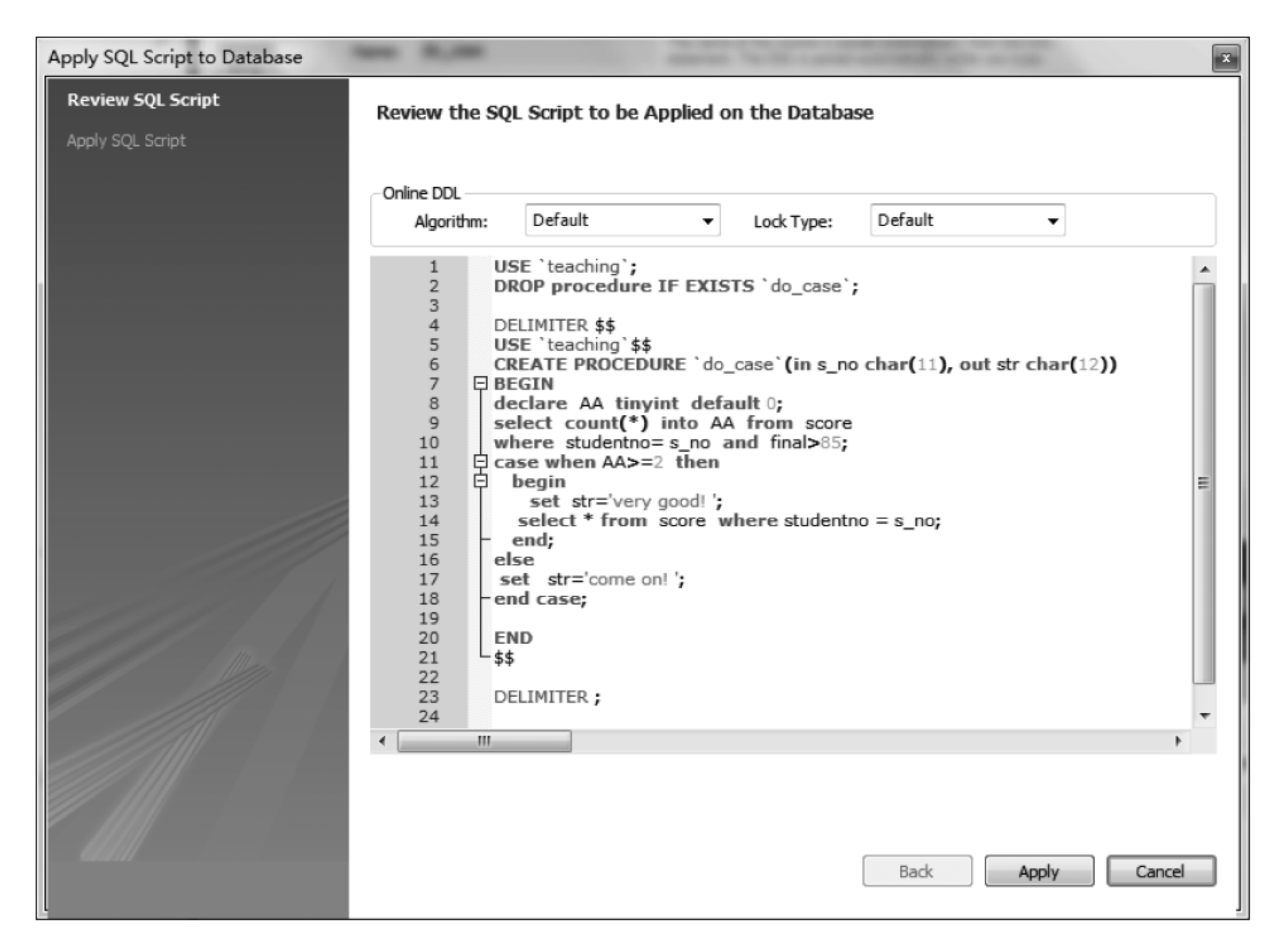


图 8-4 存储脚本

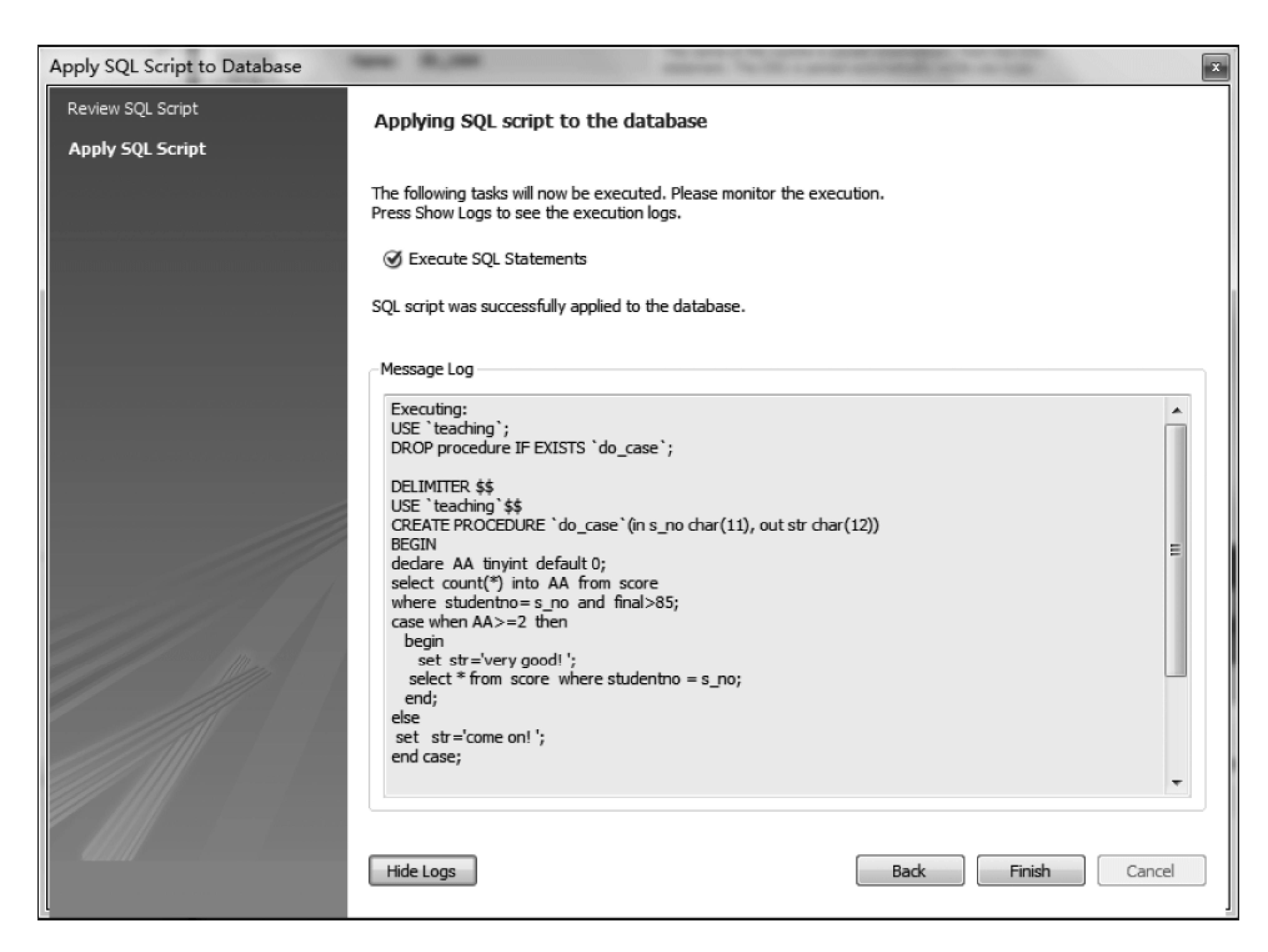


图 8-5 存储过程创建成功

3. 执行存储过程

(1) 执行存储过程 do_case，右击存储过程 do_case，选择执行 Copy to Clipboard→Procedure Call 命令，如图 8-6 所示。

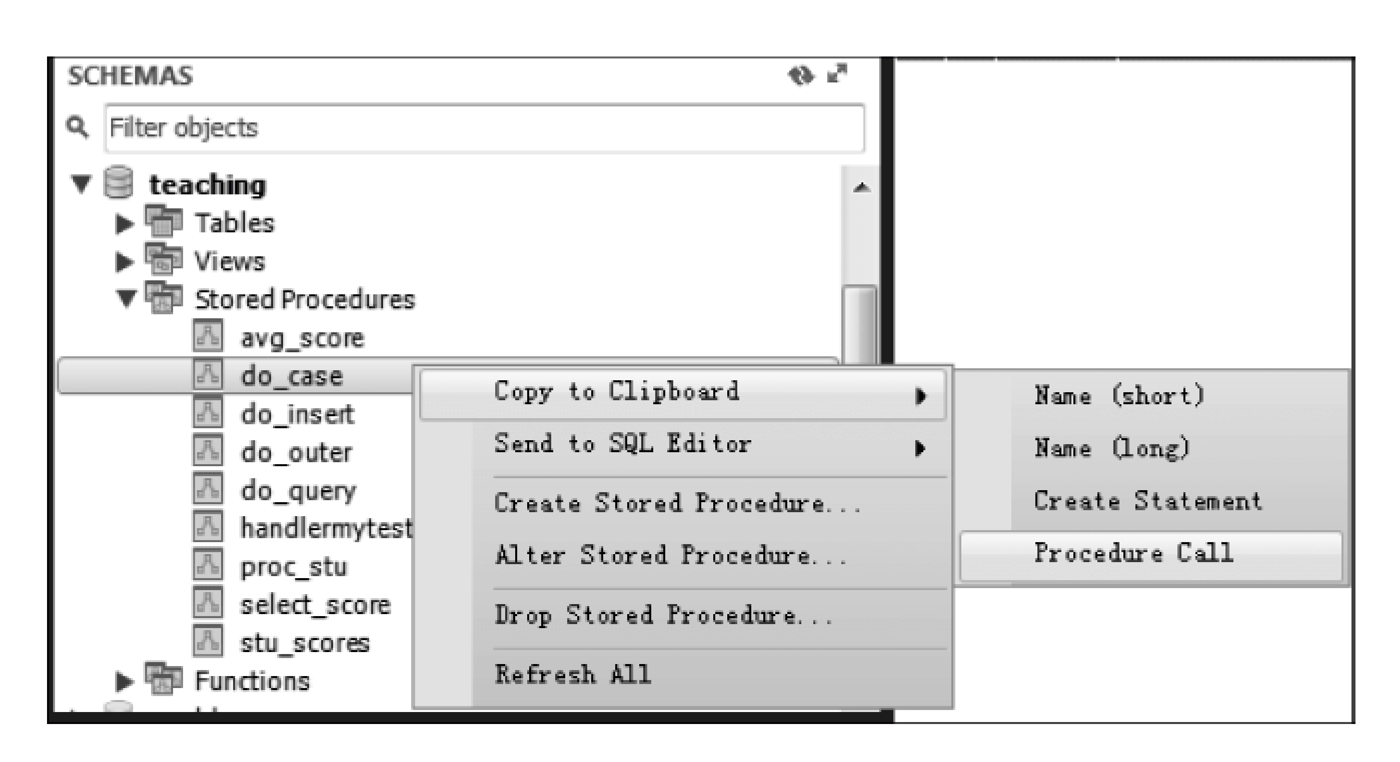


图 8-6　选择执行存储过程命令

(2) 进入执行存储过程的对话框后，在指定位置输入执行存储过程的参数，如图 8-7 所示，先输入学号 18122210009 和会话变量@str。并且可以看到工具栏中的第 3 个执行存储过程的按钮提示：执行选择的脚本参数，若未选择，就执行所有代码。

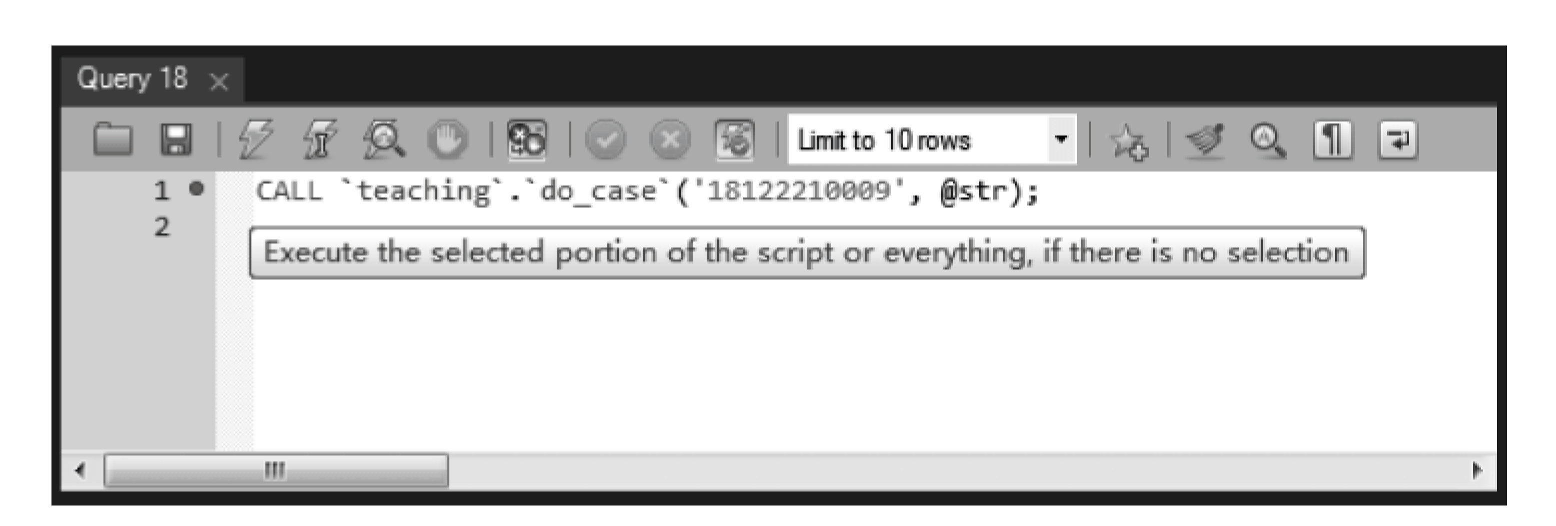


图 8-7　执行按钮提示

(3) 如图 8-8 所示，输入会话变量的执行命令“select @str”。单击执行按钮，执行结果如图 8-9 所示。

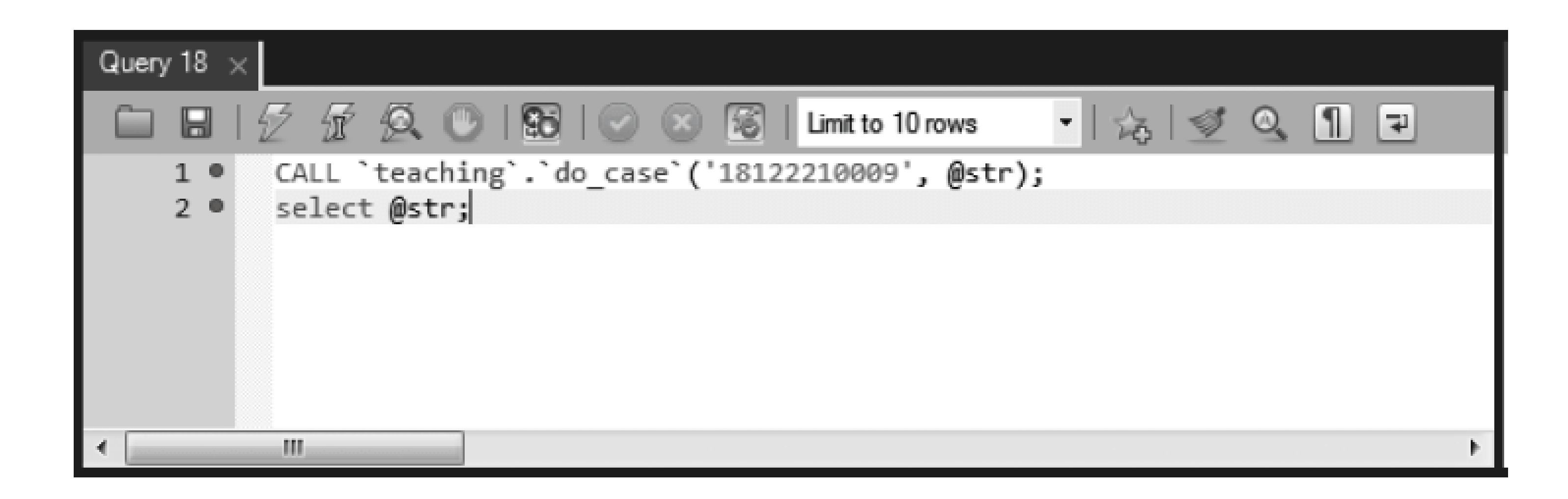


图 8-8　输入会话变量输出命令行

(4) 选择执行存储过程的对话框，再次在指定位置输入执行存储过程的参数，输入学号 18125111109 和会话变量@str。单击执行按钮，如图 8-10 所示。如果选择 Result2 选项卡，则会显示如图 8-11 所示的执行结果。由此可以看出本例与例 8-5 功能一样，对于不同的输入参数，存储过程的执行结果有时会不一样。



图 8-9　执行结果 1

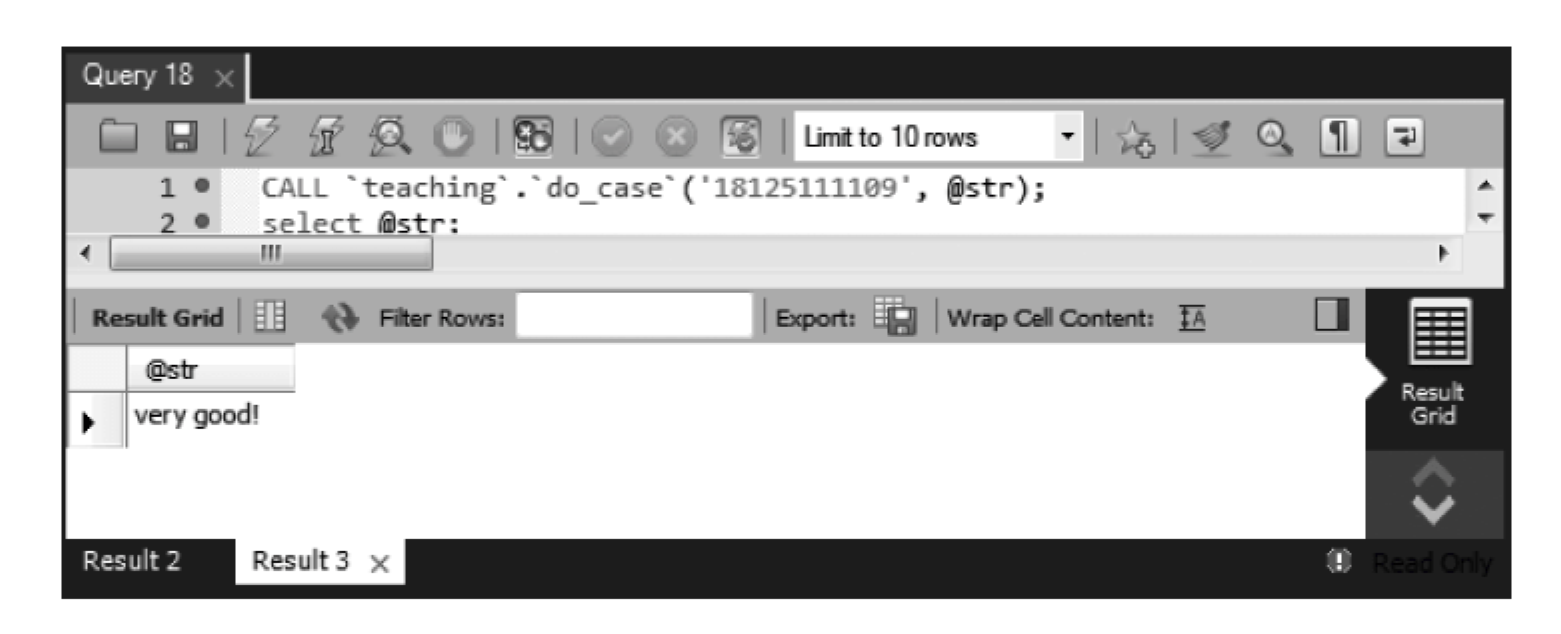


图 8-10　执行结果 2

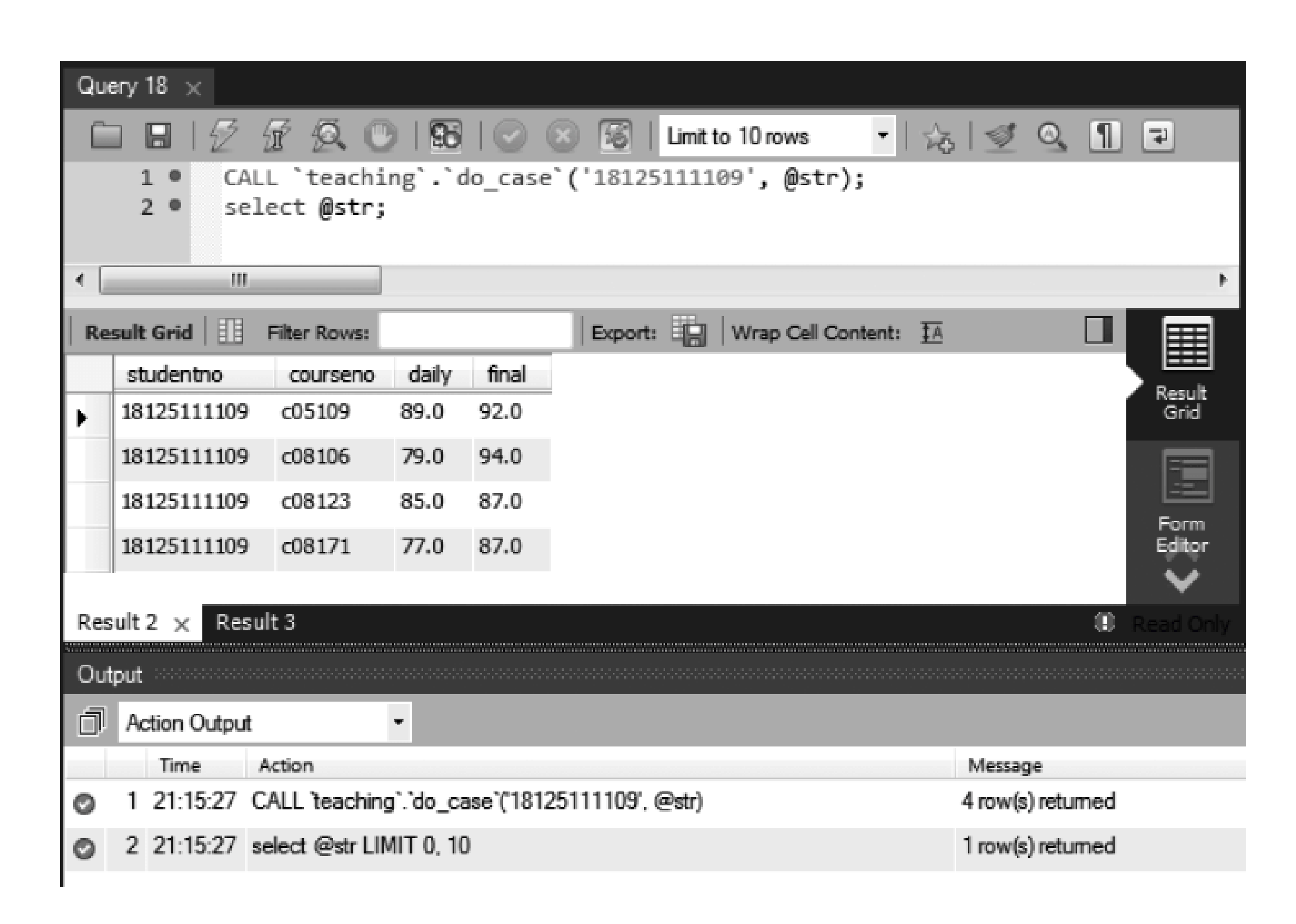


图 8-11　执行结果 3

4. 修改存储过程

（1）选择存储过程 avg_score，执行 Alter Stored Procedures 命令，如图 8-12 所示。

（2）在修改存储过程编辑框中，输入修改项，如存储过程名改为 max_score，函数 avg(final)改为 max(final)，如图 8-13 所示。

（3）认真检查代码无误后，单击 Apply 按钮，进入代码检查对话框中，向数据库 teaching 中存储脚本。再次单击 Apply 按钮，进入完成修改对话框中，单击 Finish 按钮完成存储过程的修改。



图 8-12　修改存储过程初始界面



图 8-13　修改存储过程

在利用 MySQL Workbench 工具删除存储过程时，只要对要删除的存储过程执行 Drop Stored Procedures 命令即可。

8.2　利用游标处理结果集

在 MySQL 数据库中的大部分数据管理操作都与 select 语句有关。select 语句执行后一般会产生包含多条记录的、存放在客户机内存中的结果集。数据库开发人员编写存储过程或函数等存储程序时，有时需要访问 select 结果集中的具体数据行，对结果集中的每条记录进行处理。游标(Cursor)机制就是可以解决此类问题的主要方法。



游标

游标在 MySQL 中是一种对 select 语句结果集进行访问的机制。MySQL 服务器会专门为游标开辟一定的内存空间，以存放游标操作的结果集数据，同时游标的使用也会根据具体情况对某些数据进行封锁。游标能够实现允许用户访问单独的数据行，而不是只能对整个结果集进行操作。

游标主要包括结果集和游标位置两部分，游标结果集是定义游标的 select 语句的结果集，游标位置则是指向这个结果集中的某一行的指针。

游标的使用过程如图 8-14 所示，可以概括为声明游标、打开游标、从游标中提取数据以及关闭游标。

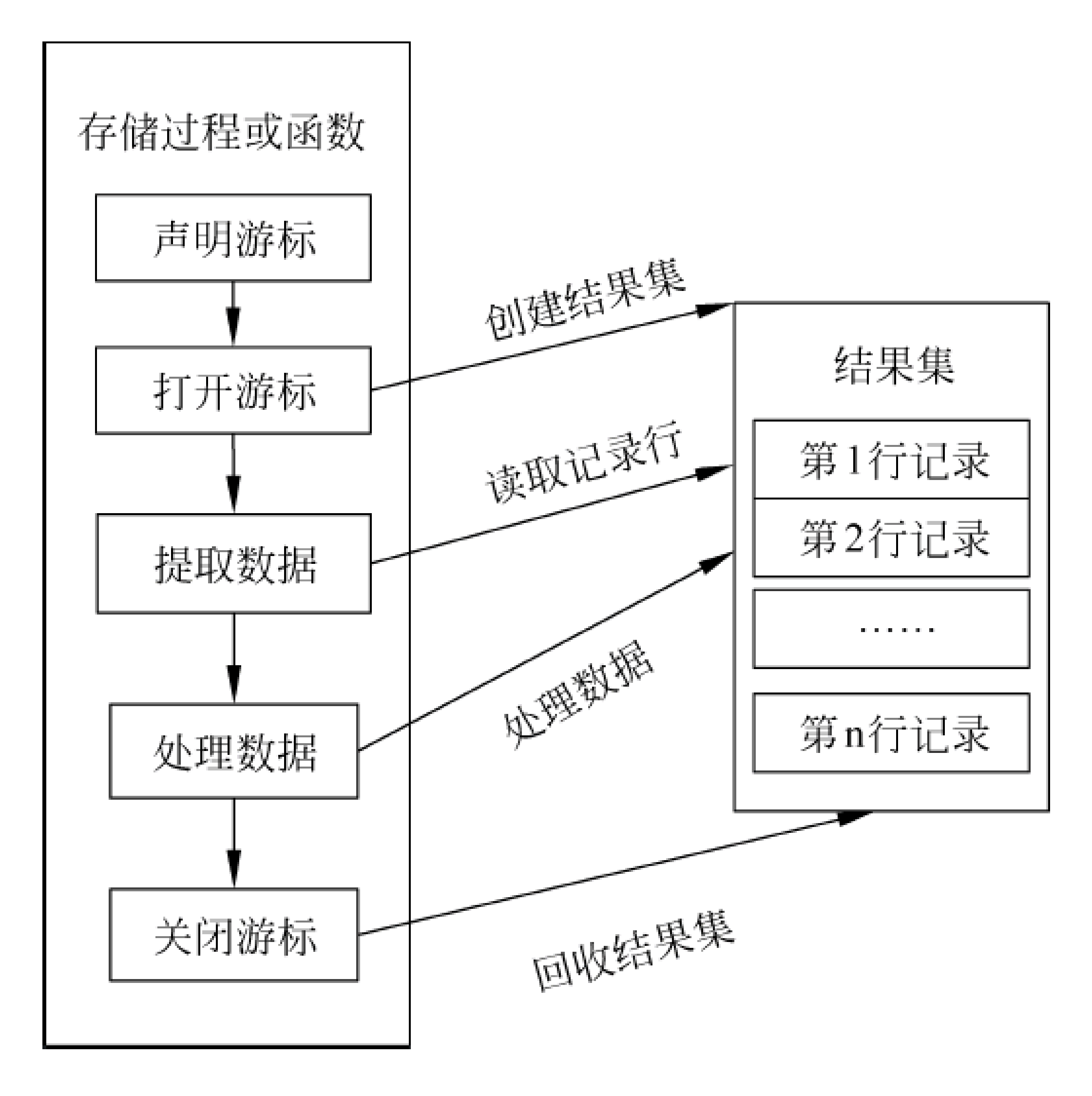


图 8-14　游标的执行过程

1. 声明游标

声明游标需要使用 declare 语句,声明游标的语法格式如下:

```
declare cursor_name cursor
for select_statement;
```

例如,在 teaching 数据库中为 teacher 表创建一个普通的游标,定义名称为 teach_cursor,声明游标 teach_cursor 的语句如下:

```
declare teach_cursor cursor
for select teacherno,tname from teacher;
```

使用 declare 语句声明游标后,此时与该游标对应的 select 语句并没有执行,MySQL 服务器内存中并不存在与 select 语句对应的结果集。

2. 打开游标

打开游标需要使用 open 语句,使用游标之前必须首先打开游标,打开游标的语法如下所示:

```
open cursor_name;
```

例如,打开前面创建的 teach_cursor 游标,使用如下语句:

```
open teach_cursor;
```

使用 open 语句打开游标后,与该游标对应的 select 语句将被执行,MySQL 服务器内存中将存放与 select 语句对应的结果集。

3. 从游标中提取数据

在打开游标以后,就可以从游标中提取数据。从游标中提取数据需要使用 fetch 语句,fetch 语句的功能是获取游标当前指针的记录,并传给指定变量列表。如果需要提取多行数据,则需要使用循环语句去执行 fetch 语句,MySQL 的游标是向前只读的,即只能顺序地从

开始往后读取结果集,不能从后往前,也不能直接跳到中间的记录。

fetch 语句的语法结构如下:

```
fetch cursor_name into var1[,var2,...];
```

说明:

(1) 变量名的个数必须与声明游标时使用的 select 语句结果集中的字段个数保持一致。第 1 次执行 fetch 语句时,fetch 语句从结果集中提取第 1 条记录,再次执行 fetch 语句时,fetch 语句从结果集中提取第 2 条记录,……以此类推。

(2) fetch 语句每次从结果集中仅仅提取一条记录,因此 fetch 语句需要循环语句的配合,才能实现整个结果集的遍历。fetch 离不开循环语句。一般使用 loop 和 while 比较清楚,而且代码简单。这里使用 loop 为例,代码如下:

```
fetchloop:loop
fetch teach_cursor into v_tno,v_tname;
end loop;
```

上述循环是死循环,没有退出的条件。MySQL 是通过一个 Error handler 的声明来进行判断的。该语句语法格式如下:

```
declare continue handler for not found … ;
```

(3) 当使用 fetch 语句从游标中提取最后一条记录后,再次执行 fetch 语句时,将产生“ERROR 1329 (02000): No data to fetch”错误信息,数据库开发人员可以针对 MySQL 错误代码 1329,自定义错误处理程序以便结束“结果集”的遍历。

(4) 游标错误处理程序应该放在声明游标语句之后。游标通常结合错误处理程序一起使用,用于结束结果集的访问。

4. 关闭游标

关闭游标使用 close 语句,关闭游标的具体语法如下:

```
close cursor_name;
```

关闭游标的目的在于释放游标打开时产生的结果集,以通知服务器释放游标所占用的资源,节省 MySQL 服务器的内存空间。游标如果没有被明确地关闭,将在它被声明的 begin...end 语句块的末尾关闭。

使用声明过的游标不需要再次声明。如果不明确关闭游标,MySQL 将会在到达 end 语句时自动关闭。

在检索游标 teach_cursor 后可用如下语句来关闭。

```
close teach_cursor;
```

【例 8-10】 创建存储过程,利用循环语句控制 fetch 语句来检索游标 teach_cursor 中可用的数据的示例。

代码和运行结果如下:

```
mysql> use teaching;
    Database changed
```

```
mysql> delimiter //
mysql> create procedure proc_cursor()
    -> begin
    -> declare v_tno varchar(6) default '';
    -> declare v_tname varchar(8) default '';
    -> declare teach_cursor cursor
    ->   for select teacherno, tname from teacher;
    -> declare continue handler for not found set @dovar = 1; #定义处理程序
    -> set @dovar  = 0;
    -> open teach_cursor;
    -> fetch_Loop:LOOP
    -> fetch teach_cursor into v_tno,v_tname;
    -> if @dovar = 1 then
    -> leave fetch_Loop;
    -> else
    -> select v_tno,v_tname;
    -> end IF;
    -> end LOOP fetch_Loop;
    -> close teach_cursor;
    -> select @dovar;
    -> end ; //
      Query OK, 0 rows affected (0.00 sec)
mysql> delimiter ;
```

调用存储过程 proc_cursor()的代码和执行结果如下：

```
mysql> call proc_cursor();
    +------+-------+
    | v_tno  | v_tname |
    +------+-------+
    | t05001 | 苏超然   |
    +------+-------+
   1 row in set (0.04 sec)
    +------+-------+
   ......
    +------+-------+
    | v_tno  | v_tname |
    +------+-------+
    | t08017 |  时观    |
    +------+-------+
   1 row in set (0.11 sec)
    +------+
    | @dovar|
    +------+
    |    1   |
    +------+
   1 row in set (0.12 sec)
   Query OK, 0 rows affected (0.12 sec)
```

利用 declare 定义一个句柄，当 fetch 抓取数据时会自动调用该句柄。如果找不到数据，会自动调用最后的 SQL 语句 set @dovar=1。其中 not found 等价于 sqlstate '02000'。

本例中，存储过程 proc_cursor()的变量@dovar 保存的就是 fetch 操作的结束信息。如果其值为零，则表示有记录检索成功，输出相应的结果；如果值为 1，则是 fetch 语句由于某种原因而操作失败。fetch 语句获取数据到结果集最后时，已经没有数据，所以执行处理程序，使得@dovar 的值为 1。

8.3 触 发 器

触发器(Trigger)是一种特殊的存储过程，可以是表定义的一部分。触发器基于一个表创建，但可以针对多个表进行操作，所以触发器可以用来对表实施复杂的完整性约束。当预定义的事件(如用户修改指定表或者视图中的数据时)发生时，触发器被自动激活，从而防止对数据进行不正确的修改。

8.3.1 认识触发器

触发器是一种特殊的存储过程，只要满足一定的条件，对数据进行 insert、update 和 delete 事件时，数据库系统就会自动执行触发器中定义的程序语句，以进行维护数据完整性或其他一些特殊的任务。如图 8-15 所示，触发器可以分为 insert、update 和 delete 三类，每一类根据执行的先后顺序又可以分成 before 和 after 触发器。

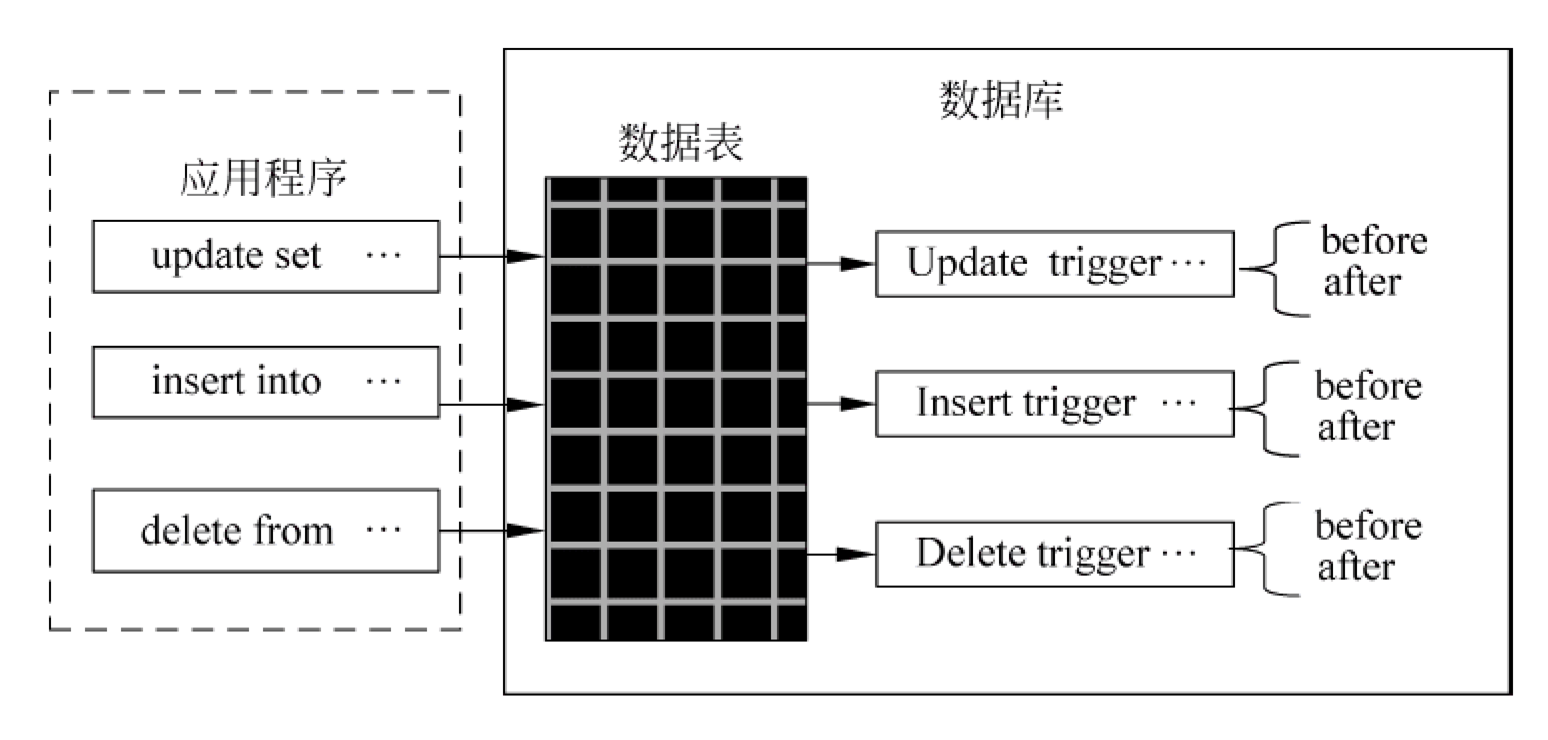


图 8-15 触发器的分类

1. 触发器的优点

(1) 触发器自动执行，在表的数据做了任何修改(例如手工输入或者使用程序采集的操作)之后立即激活。

(2) 触发器可以通过数据库中的相关表进行层叠更改。这比直接把代码写在前台的做法更安全合理。

(3) 触发器可以强制限制，这些限制比用 check 约束所定义的更复杂。与 check 约束不同的是，触发器可以引用其他表中的列。

2. 触发器的语法格式

因为触发器是一种特殊的存储过程，所以触发器的创建和存储过程的创建方式有很多相似之处。

创建触发器的语法格式如下：

```
create trigger trigger_name trigger_time trigger_event
on table_name for each row trigger_statement
```

说明：

(1) create trigger：创建触发器的关键词。触发器程序是与表有关的数据库对象，当表中出现特定事件时，将激活该对象。

(2) table_name：触发程序的相关表。table_name 必须引用永久性表。不能将触发程序与 temporary 表或视图关联起来。

(3) trigger_time：是触发程序的动作时间。可以是 before 或 after，以指明触发程序是在激活它的语句之前或之后触发。

(4) trigger_event：指明了激活触发程序的语句的类型。不支持在同一个表内同时存在两个有相同激活触发程序的类型。trigger_event 可以是下述值之一。

- insert：将新行插入表时激活触发程序。例如，通过 insert、load data 和 replace 语句。
- update：更改某一行时激活触发程序。例如，通过 update 语句。
- delete：从表中删除某一行时激活触发程序。例如，通过 delete 和 replace 语句。

(5) for each row：这个声明用来指定受触发事件影响的每一行，都要激活触发器的动作。目前 MySQL 仅支持行级触发器，不支持语句级别的触发器(例如 create table 等语句)。for each row 表示更新(insert、update 或者 delete)操作影响的每一条记录都会执行一次触发程序。

(6) trigger_statement：当触发程序激活时执行的语句。如果打算执行多个语句，可使用 begin...end 复合语句结构。这样，就能使用存储子程序中允许的相同语句。

(7) 使用触发器时，触发器执行的顺序是 before 触发器、表数据修改操作、after 触发器。其中，before 表示在触发事件发生之前执行触发程序，after 表示在触发事件发生之后执行触发器。因此严格意义上讲一个数据库表最多可以设置 6 种类型的触发器。

3. 触发程序中可以使用 old 关键字与 new 关键字

触发程序中可以使用的 old 关键字与 new 关键字实际上是在触发器事件发生时，MySQL 针对要修改数据的表，创建了与本表结构完全一样的两个临时表 old 和 new，old 表用于存放在数据修改过程中的既有数据，new 表用于存放在数据修改过程中将要更新的数据。

当向表插入新记录时，在触发程序中可以利用 new 关键字访问新记录，当需要访问新记录的某个字段值时，可以使用“new. 字段名”的方式访问。

当从表中删除旧记录时，在触发程序中可以利用 old 关键字访问旧记录，当需要访问旧记录的某个字段值时，可以使用“old. 字段名”的方式访问。

当修改表的某条记录时，在触发程序中可以使用 old 关键字访问修改前的旧记录、使用 new 关键字访问修改后的新记录。当需要访问旧记录的某个字段值时，可以使用“old. 字段名”的方式访问。当需要访问修改后的新记录的某个字段值时，可以使用“new. 字段名”的方式访问。

old 记录是只读的，只能引用，不能更改。在 before 触发程序中，可使用“set new. col_name = value”语句更改 new 记录的值。

对于 insert 语句，只有 new 是合法的；对于 delete 语句，只有 old 才合法；而 update 语句可以与 new 或 old 同时使用。

8.3.2 触发器的创建和管理



创建触发器

1. 触发器的创建和验证

触发器是由 insert、update 和 delete 等事件来触发某种特定操作。满足触发器的触发条件时，数据库系统就会执行触发器中定义的程序语句。这样做可以保证某些操作之间的一致性。

【例 8-11】 创建一个触发器，当更改表 course 中某门课的课程号时，同时将 score 表的课程号全部更新。

代码和运行结果如下：

```
mysql> use teaching;
      Database changed
mysql> delimiter $$
mysql> create trigger cno_update after update
    -> on course for each row
    -> begin
    -> update score set courseno = new.courseno
    -> where courseno = old.courseno;
    -> end $$
      Query OK, 0 rows affected (0.30 sec)
mysql> delimiter ;
```

验证触发器 cno_update 的功能，代码和执行结果如下：

```
mysql> update course set courseno  = 'c07123' where courseno = 'c08123';
      Query OK, 1 row affected (0.38 sec)
      Rows matched: 1 Changed: 1 Warnings: 0
mysql> select * from score where courseno  = 'c07123';
       +-----------+--------+-----+-----+
       | studentno   | courseno | daily | final |
       +-----------+--------+-----+-----+
       | 18125111109 | c07123   | 85.0  | 87.0  |
       | 18137221508 | c07123   | 78.0  | 84.0  |
       +-----------+--------+-----+-----+
      2 rows in set (0.00 sec)
```

说明：

(1) 在本例中，update course 是触发事件，after 是触发程序的动作时间，激发触发器 update score 表相应记录。使用 select 语句查看 score 表中的情况，发现所有原 c08123 课程编号的记录已更新为 c07123。

(2) 在 MySQL 触发器中的 SQL 语句可以关联表中的任意列。但不能直接使用列的名称标识，那会使系统混淆。

(3) 在本例中，new 和 old 同时使用。当在 course 表更新 courseno 时，原来的 courseno 变为 old. courseno，把 score 表 old. courseno 的记录要更新为 new. courseno。

【例 8-12】 在 teacher 表中，定义一个触发器，当一个教师的信息被删除时，把该教师的编号和姓名添加到 de_teacher 表中。

代码和运行结果如下：

```
#创建一个空表 de_teacher,表由 tno 和 tname 两列组成
mysql> create table de_teacher select teacherno,tname
    -> from teacher where 1 = 0;
       Query OK, 0 rows affected (0.25 sec)
       Records: 0 Duplicates: 0 Warnings: 0
#创建 teacher 表的触发器
mysql> create trigger trig_teacher
    ->  after delete on teacher for each row
    -> insert into de_teacher(teacherno,tname)
    ->   values(old.teacherno, old.tname);
     Query OK, 0 rows affected (0.06 sec)
```

验证触发器 trig_teacher 的功能，代码和执行结果如下：

```
mysql> delete from teacher where tname = '时观';
       Query OK, 1 row affected (0.08 sec)
mysql> select * from de_teacher;
       +---------+-----+
       | teacherno | tname |
       +---------+-----+
       | t08017    | 时观   |
       +---------+-----+
       1 row in set (0.00 sec)
```

2. 查看触发器的定义

既然触发器是一类特殊的存储过程，那么查看触发器是指查看数据库中已存在的触发器的定义、状态和语法信息等，也可以通过类似的命令来完成。用户可以通过 show triggers 语句来查看触发器的状态。用户也可以通过查询 information_schema 数据库下的 triggers 表来查看触发器的信息。下面给出已经验证过的查看触发器的状态和定义的方法的例子。

```
mysql> show triggers;
mysql> select * from information_schema.triggers;
mysql> select * from information_schema.triggers
    ->  where trigger_name = 'de_teacher';
```

8.3.3 使用触发器

MySQL 中的触发器在程序设计中的应用非常广泛，常见的有实现数据完整性的复杂约束、数据管理过程中的冗余数据处理以及外键约束的级联操作等，都可以利用触发器实现应用系统的自动维护。

1. 触发器应用举例

对于 InnoDB 存储引擎的表而言，由于支持外键约束，在定义外键约束时，通过设置外键的级联选项 cascade、set null 或 no action(restrict)，外键约束关系可以交由 InnoDB 存储

引擎自动维护。

【例 8-13】 创建一个触发器，当删除 student 表某个人的记录时，删除 score 表相应的成绩记录。



利用触发器实现多表级联

代码和运行结果如下：

```
mysql > delimiter $$
mysql > create trigger stu_delete after delete
    -> on student for each row
    -> begin
    -> delete from score where studentno = old.studentno;
    -> end $$
       Query OK, 0 rows affected (0.06 sec)
mysql > delimiter ;
```

验证触发器 stu_delete 的功能，代码和执行结果如下：

```
mysql > delete from student where studentno = '19112100072';
       Query OK, 1 row affected (0.12 sec)
mysql > select * from score where studentno = '19112100072';
       Empty set (0.00 sec)
```

说明：

(1) 在本例中，使用 select 语句查看 score 表中的情况，可以看到已没有 19112100072 学生的成绩记录。

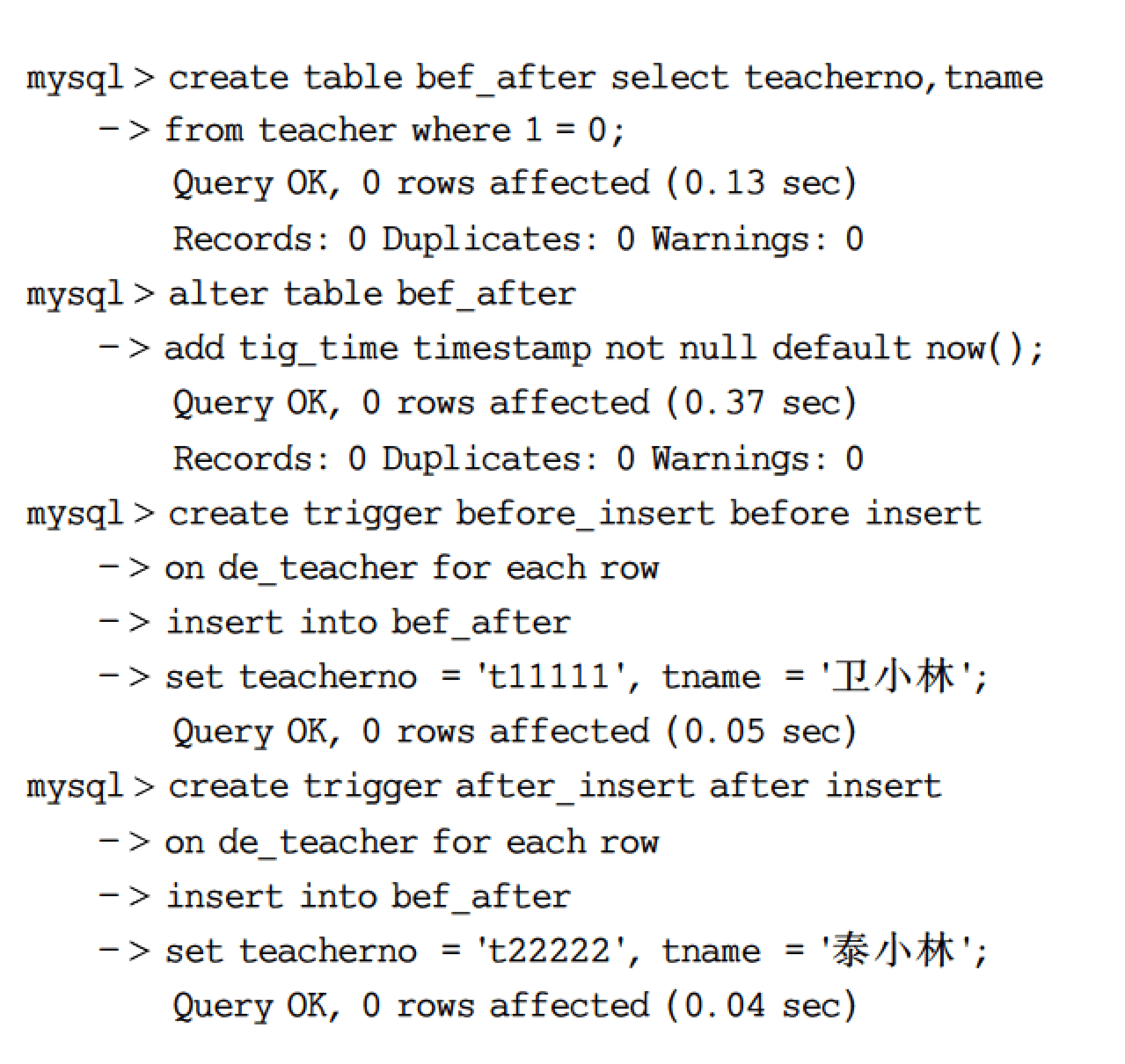
(2) 本例中，在 student 执行 delete 事件之后，在触发器中引用的 score 表的 studentno 字段要用 old.studentno 表示。

【例 8-14】 在 de_teacher 表上创建 before insert 和 after insert 这两个触发器。在向 de_teacher 表中插入数据时，观察这两个触发器的触发顺序。



触发顺序的应用

代码和运行结果如下：



```
mysql > create table bef_after select teacherno, tname
    -> from teacher where 1 = 0;
       Query OK, 0 rows affected (0.13 sec)
       Records: 0 Duplicates: 0 Warnings: 0
mysql > alter table bef_after
    -> add tig_time timestamp not null default now();
       Query OK, 0 rows affected (0.37 sec)
       Records: 0 Duplicates: 0 Warnings: 0
mysql > create trigger before_insert before insert
    -> on de_teacher for each row
    -> insert into bef_after
    -> set teacherno  = 't11111', tname  = '卫小林';
       Query OK, 0 rows affected (0.05 sec)
mysql > create trigger after_insert after insert
    -> on de_teacher for each row
    -> insert into bef_after
    -> set teacherno  = 't22222', tname  = '秦小林';
       Query OK, 0 rows affected (0.04 sec)
```

验证触发器 before_insert 和 after_insert 的功能，代码和执行结果如下：

```
mysql > insert into de_teacher values('t12345', '王含晨');
        Query OK, 1 row affected (0.05 sec)
mysql > select * from bef_after;
        +---------+------+------------------+
        | teacherno | tname  | tig_time           |
        +---------+------+------------------+
        | t11111    | 卫小林 | 2017-05-21 10:52:04 |
        | t22222    | 泰小林 | 2017-05-21 10:52:04 |
        +---------+------+------------------+
        4 rows in set (0.00 sec)
```

说明：

MySQL 中，触发器执行的顺序是 before 触发器、表操作(insert、update 和 delete)、after 触发器。本例由于程序较短，运行速度快，虽然记录的时间在 1 秒之内完成，但记录的插入顺序可以说明 before 触发器的执行早于 after 触发器。

2. 使用触发器的注意事项

(1) 触发程序中如果包含 select 语句，该 select 语句不能返回结果集。

(2) 同一个表不能创建两个相同触发时间、触发事件的触发程序。

(3) 触发程序中不能使用以显式或隐式方式打开、开始或结束事务的语句，如 start transaction、commit、rollback 或者 set autocommit=0 等语句。

(4) MySQL 触发器针对记录进行操作，当批量更新数据时，引入触发器会导致更新操作性能降低。

(5) 在 MyISAM 存储引擎中，触发器不能保证原子性。InnoDB 存储引擎支持事务，使用触发器可以保证更新操作与触发程序的原子性，此时触发程序和更新操作是在同一个事务中完成的。

(6) InnoDB 存储引擎实现外键约束关系时，建议使用级联选项维护外键数据；MyISAM 存储引擎虽然不支持外键约束关系，但可以使用触发器实现级联修改和级联删除，进而维护"外键"数据，模拟实现外键约束关系。

(7) 使用触发器维护 InnoDB 外键约束的级联选项时，数据库开发人员究竟应该选择 after 触发器还是 before 触发器？答案是应该首先维护子表的数据，然后再维护父表的数据，否则可能出现错误。

(8) MySQL 的触发程序不能对本表使用更新语句(例如 update 语句)。触发程序中的更新操作可以直接使用 set 命令替代，否则可能出现错误信息，甚至陷入死循环。

(9) 在 before 触发程序中，auto_increment 字段的 new 值为 0，不是实际插入新记录时自动生成的自增型字段值。

(10) 添加触发器后，建议对其进行详细的测试，测试通过后再决定是否使用触发器。

8.3.4 删除触发器

删除触发器指删除数据库中已经存在的触发器。MySQL 使用 drop trigger 语句来删除触发器。其基本形式如下：

```
drop trigger [schema_name.]trigger_name
```

例如，删除触发器 stu_score 的代码如下：

```
mysql > drop trigger stu_score;
```

8.4 事件及其应用

8.4.1 认识事件

MySQL 中的事件(Event)又称事件调度器(Event Scheduler)，是一种定时任务机制，可以用于定时执行诸如删除记录、对数据进行汇总等某些特定任务，来取代原先只能由操作系统的计划任务来执行的工作。

MySQL 的事件调度器可以精确到每秒钟执行一个任务，比操作系统的计划任务(如 Linux 下的 cron 或 Windows 下的任务计划)只能精确到每分钟执行一次有实时优势。对于一些对数据实时性要求比较高的应用，如股票交易、火车购票、球赛技术统计等就非常适合。一些对数据管理的定时性操作不再依赖外部程序，直接使用数据库本身提供的功能即可。

(1) 开启事件调度器。MySQL 的事件调度器是 MySQL 数据库服务器的一部分，负责调用事件，并不断地监视一个事件是否需要调用。要创建事件，必须打开调度器。

可以使用系统变量@@event_scheduler 来打开事件调度器，true(或 1 或 on)为打开，false(或 0 或 off)为关闭。

要开启 event_scheduler，可执行下面的语句。

```
set @@global.event_scheduler = true;
```

也可以在 MySQL 的配置文件 my.ini 中加上一行，然后重启 MySQL 服务器。

```
event_scheduler = 1
```

(2) 查看事件调度器。要查看当前是否已开启事件调度器，可执行如下相关 SQL 语句。

代码和运行结果如下：

```
mysql > set @@global.event_scheduler = true;
        Query OK, 0 rows affected (0.05 sec)
mysql > show variables like 'event_scheduler';
         +--------------+-----+
         | Variable_name   | Value |
         +--------------+-----+
         | event_scheduler |  on   |
         +--------------+-----+
         1 row in set, 1 warning (0.00 sec)
mysql > select @@ event_scheduler;
         +-----------------+
         | @@event_scheduler |
         +-----------------+
```

```
| on              |
+-----------------+
1 row in set (0.00 sec)
```

8.4.2 创建事件



创建事件

创建事件可以创建在某一时刻发生的事件、指定区间周期性发生的事件，以及在事件中调用存储过程或存储函数的实际应用。

1. 创建事件的一般格式

创建事件可以使用 create event 语句，语法格式如下：

```
create event [if not exists] event_name
on schedule schedule
[on completion [not] preserve]
[enable|disable|disable on slave]
[comment'comment']
do sql_statement;
```

其中：

```
schedule: at timestamp [ + interval interval]
        |every interval[starts timestamp [ +  interval interval]]
        [ends timestamp[ +  interval interval]]
interval: count { year|quarter|month|day|hour|minute
        |week|second|year_month|day_hour|day_minute
        |day_second|hour_minute|hour_second|minute_second}
```

说明：

(1) event_name：表示事件名。

(2) schedule：是时间调度，表示事件何时发生或者每隔多久发生一次。

- at 子句：表示事件在某个时刻发生。timestamp 表示一个具体的时间点，后面还可以加上一个时间间隔，表示在这个时间间隔后事件发生。interval 表示这个时间间隔，由一个数值和单位构成，count 是间隔时间的数值。
- every 子句：表示在指定时间区间内每隔多长时间事件发生一次。starts 子句指定开始时间，ends 子句指定结束时间。

(3) do sql_statement：事件启动时执行的 SQL 代码。如果包含多条语句，可以使用 begin...end 复合结构。

2. 创建某个时刻发生的事件

【例 8-15】 创建现在立刻执行的事件 direct1，创建一个表 test1。

代码和运行结果如下：

```
mysql > use mysqltest;
       Database changed
mysql > create event direct1
    -> on schedule at now()
    -> do
    -> create table test1(timeline timestamp);
```

```
    Query OK, 0 rows affected (0.00 sec)
mysql> show tables;
    +-----------------+
    | Tables_in_mysqltest |
    +-----------------+
    | course01        |
    | score1          |
    | student01       |
    | student02       |
    | teacher1        |
    | test1           |
    +-----------------+
    6 rows in set (0.00 sec)
mysql> select * from test1;
    Empty set (0.00 sec)
```

【例 8-16】 创建现在立刻执行的事件 direct2,5 秒后创建一个表 test2。

代码和运行结果如下：

```
mysql> create event direct2
    -> on schedule at current_timestamp + interval 5 second
    -> do
    -> create table test2(timeline timestamp);
       Query OK, 0 rows affected (0.07 sec)
```

3. 创建在指定区间周期性发生的事件



周期性事件

【例 8-17】 创建事件 test1_insert,每秒插入一条记录到数据表 test1。

代码和运行结果如下：

```
mysql> create event test1_insert
    -> on schedule every 1 second
    -> do
    -> insert into test1 values (current_timestamp);
       Query OK, 0 rows affected (0.00 sec)
mysql> select * from test1; #5 秒之后执行此语句
     +------------------+
     | timeline         |
     +------------------+
     | 2017-05-22 18:12:18 |
     | 2017-05-22 18:12:19 |
     | 2017-05-22 18:12:20 |
     | 2017-05-22 18:12:21 |
     | 2017-05-22 18:12:22 |
     +------------------+
     5 rows in set (0.02 sec)
```

【例 8-18】 创建事件 startweeks,要求从下周开始,每周都清空 test1 表,并且在 2017 年的 08 月 31 日 12:00 结束。

代码和运行结果如下：

```
mysql> delimiter $$
```

```
mysql > create event startweeks
    -> on schedule every 1 week
    -> starts curdate() + interval 1 week
    -> ends '2017-08-31 12:00:00'
    -> do
    -> begin
    ->   truncate table test1;
    -> end $$
       Query OK, 0 rows affected (0.00 sec)
mysql > delimiter;
```

4. 在事件中调用存储过程或存储函数


利用事件调用过程

【例 8-19】 存储过程 proc_stu()用于查询学生信息的,创建事件 stu_week 每周查看一次学生的情况。

代码和运行结果如下:

```
mysql > delimiter $$
mysql > create event stu_week
    -> on schedule every 1 week
    -> do
    -> begin
    -> call teaching.proc_stu();
    -> end $$
       Query OK, 0 rows affected (0.03 sec)
mysql > delimiter;
```

8.4.3 管理事件

1. 查看事件

(1) MySQL 中查看所有事件 event 的语法如下:

```
show events [from schema_name]
[like 'pattern'|where expr]
```

可以直接使用命令"show events;"查看数据库 mysqltest 中的事件。为了直观一些,采用如下的方法查看。

【例 8-20】 格式化显示所有事件 event。

代码和运行结果如下:

```
mysql > show events\G
        *************************** 1. row ***************************
                     Db: mysqltest
                   Name: startweeks
                Definer: root@localhost
              Time zone: SYSTEM
                   Type: RECURRING
             Execute at: NULL
         Interval value: 1
         Interval field: week
```

```
                    Starts: 2017-05-29 00:00:00
                      Ends: 2017-08-31 12:00:00
                    Status: enableD
                Originator: 1
      character_set_client: gbk
      collation_connection: gbk_chinese_ci
        Database Collation: gbk_chinese_ci
************************** 2. row **************************
                        Db: mysqltest
                      Name: stu_week
          …
************************** 3. row **************************
                        Db: mysqltest
                      Name: test1_insert
                   Definer: root@localhost
           …
3 rows in set (0.02 sec)
```

(2) MySQL 中查看 event 的创建信息的语法如下：

```
mysql> show create event event_name;
```

例如，查看 stu_week 的创建信息的代码如下：

```
mysql> show create event stu_week;
```

2. 修改事件

MySQL 中可以通过 alter event 语句来修改事件的定义和相关属性。具体修改格式如下：



修改事件

```
alter event event_name
[on schedule schedule]
[rename to new_event_name]on completion[not]preserve]
[comment'comment'][enable|disable][do sql_statement]
```

例如，可以临时关闭事件或再次让它活动，修改事件的名称并加上注释等。

【例 8-21】 对事件 test1_insert 进行操作如下：临时关闭 test1_insert 事件；开启 test1_insert 事件，将每天清空 test1 表改为 7 天清空一次；重命名事件 test1_insert 并加上注释。

代码和运行结果如下：

```
mysql> alter event test1_insert disable;
       Query OK, 0 rows affected (0.00 sec)
mysql> alter event test1_insert enable;
       Query OK, 0 rows affected (0.00 sec)
mysql> alter event test1_insert on schedule every 7 day;
       Query OK, 0 rows affected (0.00 sec)
mysql> alter event test1_insert
    -> rename to insert_test1 comment '表 test1 的数据操作';
       Query OK, 0 rows affected (0.00 sec)
```

3. 删除事件

MySQL 中用 drop event 删除事件，删除事件的语法格式如下：

```
drop event [if exists][database name.]event_name
```

例如，删除事件 insert_test1 的代码如下：

```
mysql > drop event insert_test1;
```

8.5 小　结

本章介绍了 MySQL 数据库的存储过程，以及利用存储过程实现的游标和事件的创建、应用和管理。而触发器则是一种能够自动执行的特殊存储过程。存储过程和存储函数都是用户自己定义的 SQL 语句的集合。它们都存储在服务器端，只要调用就可以在服务器端执行。本章还介绍了 MySQL 数据库的触发器的定义和作用、创建触发器、查看触发器、使用触发器、删除触发器等内容。学习本章后应该重点掌握如下内容：

- 存储过程的创建和使用，存储过程和存储函数的区别。
- 触发器的创建和使用，利用触发器能够实现哪些操作。
- 事件的创建过程和使用场合。
- 游标的执行过程，如何利用游标访问结果集的步骤。

习　题　8

1. 选择题

(1) 存储过程是在 MySQL 服务器中定义并________的 SQL 语句集合。

A. 保存　　B. 执行　　C. 解释　　D. 编写

(2) 下面有关存储过程的叙述错误的是________。

A. MySQL 允许在存储过程创建时引用一个不存在的对象

B. 存储过程可以带多个输入参数，也可以带多个输出参数

C. 使用存储过程可以减少网络流量

D. 在一个存储过程中不可以调用其他存储过程

(3) MySQL 所支持的触发器不包括________。

A. insert 触发器　　B. delete 触发器　　C. check 触发器　　D. update 触发器

(4) 下面有关触发器的叙述错误的是________。

A. 触发器是一个特殊的存储过程

B. 触发器不可以引用所在数据库以外的对象

C. 在一个表上可以定义多个触发器

D. 触发器在 check 约束之前执行

(5) MySQL 为每个触发器创建了两个临时表________。

A. max 和 min　　B. avg 和 sum　　C. int 和 char　　D. old 和 new

(6) 通过以下________语句临时关闭事件 e_test。

A. alter event e_test disable　　B. alter event e_test drop

C. alter event e_test enable　　D. alter event e_test delete

（7）下列________语句用来定义游标。

A. create　　B. declare

C. declare...cursor for...　　D. show

（8）下列说法中错误的是________。

A. 常用触发器有 insert、update、delete 三种

B. 对于同一张数据表，可以同时有两个 before update 触发器

C. new 临时表在 insert 触发器中用来访问被插入的行

D. old 临时表中的值只读，不能被更新

（9）存储程序中的选择语句有________。

A. if　　B. while　　C. select　　D. switch

（10）存储程序中不能使用的循环语句是________。

A. repeat　　B. while　　C. loop　　D. for

2. 思考题

（1）MySQL 中创建多条执行语句的存储过程或触发器时，为何总是遇到分号就结束创建，然后报错？如何解决这个问题？

（2）各种触发器的触发顺序是什么？

（3）什么是事件？事件有什么作用？事件与触发器的区别有哪些？

（4）简述游标在存储过程中的作用。

（5）简述存储过程与存储函数的区别。

3. 上机练习题（以下题目默认数据库为 teaching）

（1）创建存储过程 selectscore()，用指定的学号查询学生成绩。

（2）编程在表 course 中创建一个触发器 course_detrigger，用于每次当删除表 course 中一行数据时，将会话变量 per1 的值设置为“old course deleted!”。

（3）创建一个存储过程，用于实现给定表 student 中一个学生的姓名即可修改表 student 中该学生的电子邮件地址为一个给定的值。

（4）创建一个事件，用于每 6 个月将表 score 中期末成绩高于 60 分的所有记录信息删除，该事件开始于下个月并且在 2019 年 12 月 31 日结束。

（5）创建一个存储过程 scoreInfo，完成的功能是在表 student、表 course 和表 score 中查询以下字段：学号、姓名、性别、课程名称、期末分数。

（6）创建一个带有参数的存储过程 stu_age，该存储过程根据输入的学号，在 student 表中计算此学生的年龄，并根据程序的执行结果返回不同的值，程序执行成功，返回整数 0，如果执行出错，则返回错误号。

（7）创建事件 e_test，每天定时清空 test 表，5 天后停止执行。

（8）假设之前创建的 course 表没有设置外键级联策略，设置触发器，实现在 course 表中删除课程信息时，可自动删除该课程在 score 上的成绩信息。

第9章　并发事务与锁机制

MySQL在对数据库进行操作时，通过事务来保证数据的完整性。事务由一系列的数据操作命令序列组成，是数据库应用程序的基本逻辑操作单元。在MySQL环境中，事务由作为一个逻辑单元的一个或多个SQL语句组成。例如，前面介绍的每一条DDL语句，都可以看成是一个事务；但在实际的工作中，一个事务往往是需要多条语句共同组成，来完成较为复杂的数据操作。

多用户访问数据库时，并发的情况是常态，数据库系统的并发处理能力是衡量其性能的重要标志之一。数据库系统需要通过适当的并发控制机制协调并发操作，保证数据的一致性。在MySQL数据库中，事务是进行数据管理的基本操作单元，锁机制是用于实现并发控制的主要方法。

本章主要介绍事务与锁的基本概念和基本操作。

9.1　认识事务机制

在程序设计过程中，与一个事务相关的数据必须保证可靠性、精确性、一致性和完整性，以符合实际的企业生产过程的需要。现实生活中如火车购票、网上购物、股票交易、银行借贷等都是采用事务方式来处理的。

在MySQL中，通常由事务来完成相关操作，以确保多个数据的修改作为一个单元来处理。例如，银行系统的转账业务是最基本的、且最常用的业务，有必要将转账业务的数据操作命令封装成含有事务的存储过程，调用该存储过程后即可实现两个银行账户间的数据的可靠性和完整性转账。在银行存贷业务中有一条记账原则，即“有借有贷，借贷相等”。为了保证这条原则，就得确保“借”和“贷”的登记要么同时成功，要么同时失败。如果出现了只记录“借”，或者只记录“贷”的情况，就违反了记账原则，通常称为“记错账”，数据的可靠性和完整性就无法保证。而此过程实际上就是对银行服务器中的数据表进行了含有一组数据修改的SQL语句的事务操作。

9.1.1　事务的特性

事务处理机制在程序开发过程中有着非常重要的作用，它可以使整个系统更加安全。MySQL系统具有事务处理功能，能够保证数据库操作的一致性和完整性，使用事务可以确保同时发生的行为与数据的有效性不发生冲突。在MySQL中，并不是所有的存储引擎都支持事务，如InnoDB和BDB支持，但MyISAM和MEMORY则不支持。

事务中的每个SQL语句是互相依赖的，而且单元作为一个整体是不可分割的。如果单

元中的一个语句不能完成，整个单元就会回滚全部数据操作，返回到事务开始以前的状态。因此，只有事务中的所有语句都执行完毕，才能说这个事务被成功地执行，才能将执行结果提交到数据库文件中，成为数据库永久的组成部分。因为由用户并发访问数据库引发的数据操作经常会同时发生在多个数据表上，为了能够保证数据的一致性，必须要求这些操作不能发生中断。这就要求事务本身必须具有以下 4 个特性。

(1) 原子性(Atomicity)。原子性意味着每个事务都必须被看作一个不可分割的单元。假设一个事务由两个或者多个任务组成，其中的语句必须同时成功才能认为整个事务是成功的。如果事务失败，系统将会返回到该事务开始执行前的状态。

(2) 一致性(Consistency)。事务执行完成后，都将数据库从一个一致状态转变到另一个一致状态，事务不能违背定义在数据库中的任何完整性检查。一致性在逻辑上不是独立的，它由事务的隔离性来表示。

(3) 隔离性(Isolation)。隔离性是指每个事务在其自己的会话空间发生，和其他发生在系统中的事务隔离，而且事务的结果只有在完全被执行后才能看到。即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰。该机制是通过对事务的数据访问对象加适当的锁，排斥其他事务对同一数据库对象的并发操作来实现的。

(4) 持久性(Durability)。要求一旦事务提交，那么对数据库所做的修改将是持久的，无论发生何种机器和系统故障，都不应该对其有任何影响。大多数 DBMS 产品通过保存所有行为的日志来保证数据的持久性，这些行为是指在数据库中以任何方法更改数据。数据库日志记录了所有对于表的更新、查询、报表等。例如，自动柜员机(ATM)在向客户支付一笔钱时，只要操作提交，就不用担心丢失客户的取款记录。

9.1.2 事务的分类

任何对数据的修改都是在事务环境中进行的。按照事务定义的方式可以将事务分为系统定义事务和用户定义事务。MySQL 支持 4 种事务模式分别对应上述两类事务，自动提交事务、显式事务、隐式事务和适合多服务器系统的分布式事务。其中显式事务和隐式事务属于用户定义的事务。

(1) 自动提交事务。默认情况下，MySQL 采用 autocommit 模式运行。当执行一个用于修改表数据的语句之后，MySQL 会立刻将结果存储到磁盘中。如果没有用户定义事务，MySQL 会自己定义事务，称为自动提交事务。每条单独的语句都是一个事务。例如，InnoDB 中的 create table 语句被作为一个单一事务进行处理。即用户执行 rollback 语句不会回滚用户在事务处理过程中创建的 create table 语句。

每个 MySQL 语句在完成时，都被提交或回滚。如果一个语句成功地完成，则提交该语句。如果遇到错误，则回滚该语句的操作。只要没有显式事务或隐式事务覆盖自动提交模式，与数据库引擎实例的连接就以此默认模式操作。

(2) 用户定义事务。显式事务是指显式定义了启动(start transaction | begin work)和结束(commit 或 rollback work)的事务。在实际应用中，大多数的事务是由用户来定义的。事务结束分为提交(commit)和回滚(rollback)两种状态。事务以提交状态结束，全部事务操作完成后，将操作结果提交到数据库中。事务以回滚的状态结束，则将事务的操作全部取

消，事务操作失败。而隐式事务则不需要定义启动和结束等操作，而是由一些 MySQL 语句隐式地执行相关操作。

(3) 分布式事务。一个比较复杂的环境，可能有多台服务器，那么要保证在多服务器环境中事务的完整性和一致性，就必须定义一个分布式事务。在分布式事务中，所有的操作都可以涉及对多个服务器的操作，当这些操作都成功时，那么所有这些操作都提交到相应服务器的数据库中，如果这些操作中有一条操作失败，那么这个分布式事务中的全部操作都被取消。

InnoDB 存储引擎支持 XA 事务，通过 XA 事务可以支持分布式事务的实现。XA 协议作为资源管理器（数据库）与事务管理器的接口标准。目前，Oracle、Informix、DB2 和 Sybase 等各大数据库厂家都提供对 XA 的支持。

分布式事务指的是允许多个独立的事务资源（Transactional Resources）参与一个全局的事务。事务资源通常是关系型数据库系统，也可以是其他类型的资源。

全局事务要求在其中所有参与的事务要么全部提交，要么全部回滚，这对于事务原有的 ACID 要求又有了提高。另外，在使用分布式事务时，InnoDB 存储引擎的事务隔离级别必须设置成 serialiable。

XA 事务允许不同数据库之间的分布式事务，如一台服务器是 MySQL 数据库，一台是 Oracle 数据库，可能还有一台是 MySQL 数据库，只要参与全局事务中的每个节点都支持 XA 事务。分布式事务可能在银行系统的转账中比较常见。

分布式事务是由一个或多个 Resource Managerd，一个事务管理器 Transaction Manager 以及一个应用程序 Application Program 组成。

- 资源管理器：提供访问事务资源的方法，通常一个数据库就是一个资源管理器。
- 事务管理器：协调参与全局事务中的各个事务。需要和参与全局事务中的资源管理器进行通信。
- 应用程序：定义事务的边界，指定全局事务中的操作。

在 MySQL 的分布式事务中，资源管理器就是 MySQL 数据库，事务管理器为连接到 MySQL 服务器的客户端。

分布式事务使用两段式提交（two-phase commit）的方式。在第一个阶段，所有参与全局事务的节点都开始准备，告诉事务管理器准备好提交了。第二个阶段，事务管理器告诉资源管理器执行 rollback 或者 commit，如果任何一个节点显示不能 commit，那么所有的节点就得全部 rollback。

跨越两个或多个数据库的单个数据库引擎实例中的事务实际上也是分布式事务。该实例对分布式事务进行内部管理；对于用户而言，其操作就像本地事务一样。

对于应用程序而言，分布式提交必须由事务管理器管理，以尽量避免出现因网络故障而导致事务由某些资源管理器成功提交，另一些资源管理器回滚的情况。通过准备阶段和提交阶段管理提交进程可避免这种情况，这称为两阶段提交。

9.2 事务的管理


并发事务管理

一般来说，事务的基本操作包括关闭自动提交、启动、保存、提交或回滚等环节。在 MySQL 中，当一个会话开始时，系统变量@@autocommit

值为 1，即自动提交功能是打开的，当用户每执行一条 SQL 语句后，该语句对数据库的修改就立即被提交成为持久性修改保存到磁盘上，一个事务也就结束了。因此，用户必须关闭自动提交，事务才能由多条 SQL 语句组成，可以使用如下语句来实现：

```
set @@autocommit = 0;
```

执行此语句后，必须明确地指示每个事务的终止，事务中的 SQL 语句对数据库所做的修改才能成为持久化修改。

1. 启动事务

当一个应用程序的第一条 SQL 语句或者在 commit 或 rollback 语句后的第一条 SQL 执行后，一个新的事务也就开始了。另外还可以使用一条 start transaction 语句来显式地启动一个事务。

启动事务的语法格式如下：

```
start transaction|begin work
```

利用 begin work 语句可以用来替代 start transaction 语句，但是 start transaction 更常用些。

2. 结束事务

commit 语句是提交语句，它使得自从事务开始以来所执行的所有数据修改成为数据库的永久部分，也标志一个事务的结束。

结束事务的语法格式如下：

```
commit [work][and[no]chain][[no] release]
```

注意：MySQL 使用的是平面事务模型，因此嵌套的事务是不允许的。在第一个事务里使用 start transaction 命令后，当第二个事务开始时，自动地提交第一个事务。同样，下面的这些 MySQL 语句运行时都会隐式地执行一个 commit 命令。

```
drop database / drop table /create index/ drop index/alter table / rename table /lock tables /
unlock tables /set @@autocommit = 1
```

3. 回滚事务

rollback 语句是回滚语句，它回滚事务所做的修改，并结束当前这个事务。

回滚事务的语法格式如下：

```
rollback [work][and[no]chain][[no]release]
```

在前面的举例中，若在最后加上以下这条语句：

```
rollback work;
```

执行完这条语句后，前面的删除动作将被回滚，可以使用 select 语句查看该行数据是否还原。

4. 设置事务检查点

除了回滚整个事务，用户还可以使用 rollback to 语句使事务回滚到某个点，实现事务的部分回滚。这需要使用 savepoint 语句来设置一个保存点。

设置事务检查点的语法格式如下：

```
savepoint identifier
```

其中，identifier 为保存点的名称。

利用 rollback to savepoint 语句会向已命名的保存点回滚一个事务。如果在保存点被设置后，当前事务对数据进行了更改，则这些更改会在回滚中被回滚，语法格式为：

```
rollback [work] to savepoint identifier
```

当事务回滚到某个保存点后，在该保存点之后设置的保存点将被删除。

release savepoint 语句会从当前事务的一组保存点中删除已命名的保存点。不出现提交或回滚。如果保存点不存在，会出现错误。语法格式为：

```
release savepoint identifier
```

5. 改变 MySQL 的自动提交模式

关闭自动提交的方法有两种：一种是显式地关闭自动提交，一种是隐式地关闭自动提交。

（1）显式地关闭自动提交。使用 MySQL 命令“set @@autocommit=0;”，可以显式地关闭 MySQL 自动提交。

【例 9-1】 变量@@autocommit 自动提交模式的修改示例。删除课程号为 c05103 的表记录，然后回滚。

代码和运行结果如下：

```
mysql> use teaching;
       Database changed
mysql> delimiter //
mysql> set @@autocommit = 0;
    -> create procedure auto_cno()
    -> begin
    -> start transaction;
    -> delete from course where courseno = 'c05103';
    -> select * from course where courseno = 'c05103';
    -> rollback;
    -> select * from course where courseno = 'c05103';
    -> end//
       Query OK, 0 rows affected (0.04 sec)
       Query OK, 0 rows affected (0.47 sec)
```

调用存储过程 auto_cno()，查看事务的执行结果如下：

```
mysql> call auto_cno();
       Empty set (0.50 sec)
        +--------+--------+----+------+---+----+
        | courseno | cname    | type | period | exp | term |
        +--------+--------+----+------+---+----+
        | c05103   | 高等数学 | 必修 |  64    | 16  |  2   |
        +--------+--------+----+------+---+----+
```

```
1 row in set (0.72 sec)
Query OK, 0 rows affected (0.73 sec)
```

从执行结果中发现，表 course 中已经删去课程号为 c05103 的行，显示为空记录。但是，这个修改并没有持久化，因为自动提交已经关闭了。通过 rollback 回滚这一修改，再查询时，数据回滚到了删除之前的状态。也可以使用 commit 语句持久化这一修改。

若想恢复事务的自动提交功能，执行如下语句即可：

```
set @@autocommit = 1;
```

（2）隐式地关闭自动提交。使用 MySQL 命令“start transaction;”可以隐式地关闭自动提交。隐式地关闭自动提交，不会修改系统会话变量@@autocommit 的值。

下面通过例题进一步学习事务的操作。

【例 9-2】 将 teaching 数据库的 course 表中课程号为 c05103 的课程名称改为“高等数学”，并提交该事务。



事务的应用

代码和运行结果如下：

```
mysql > use teaching;
        Database changed
mysql > delimiter //
mysql > create procedure update_cno()
    -> begin
    -> start transaction;
    -> update course set cname = '高等数学'
    -> where courseno = 'c05103';
    -> commit;
    -> select * from course where courseno = 'c05103';
    -> end//
        Query OK, 0 rows affected (0.30 sec)
mysql > delimiter ;
```

调用存储过程 update_cno()，查看事务的执行结果如下：

```
mysql > call update_cno();
        +--------+--------+----+------+---+----+
        | courseno | cname    | type | period | exp | term |
        +--------+--------+----+------+---+----+
        | c05103   | 高等数学 | 必修 |   64   | 16 |  2   |
        +--------+--------+----+------+---+----+
        1 row in set (0.80 sec)
        Query OK, 0 rows affected (0.82 sec)
```

本例中使用 start transaction 定义了一个事务，使用 commit 提交事务。执行该事务后，课程号为 c05103 的课程名称为“高等数学”。

【例 9-3】 使用显式事务向表 course 中插入两条记录。

代码和运行结果如下：

```
mysql > delimiter //
mysql > create procedure insert_cno()
```

```
    -> begin
    -> start transaction;
    -> insert into course
    ->  values('c05141','WIN设计','选修',48,8,8);
    -> insert into course
    ->  values('c05142','WEB语言','选修',32,8,8);
    -> select * from course where term = 8;
    -> commit;
    -> end//
       Query OK, 0 rows affected (0.01 sec)
mysql> delimiter ;
```

调用存储过程 insert_cno()，查看事务的执行结果如下：

```
mysql> call insert_cno();
       +--------+---------+-----+------+----+-----+
       | courseno | cname    | type | period | exp | term |
       +--------+---------+-----+------+----+-----+
       | c05141  | WIN设计  | 选修 |   48 |  8 |  8  |
       | c05142  | WEB语言  | 选修 |   32 |  8 |  8  |
       | c08171  | 会计软件 | 选修 |   32 |  8 |  8  |
       +--------+---------+-----+------+----+-----+
       3 rows in set (0.05 sec)
       Query OK, 0 rows affected (0.07 sec)
```

【例 9-4】 定义一个事务，向 course 表中添加一条记录，并设置保存点。然后再删除该记录，并回滚到事务的保存点，提交事务。



事务检查点

代码和运行结果如下：

```
mysql> delimiter //
mysql> create procedure sp_cno()
    -> begin
    -> start transaction;
    -> insert into course
    -> values('c05139','建模UML','选修',48,12,7);
    -> savepoint spcno1;
    -> delete from course
    -> where courseno = 'c05139';
    -> rollback work to savepoint spcno1;
    -> select * from course where courseno = 'c05139';
    -> commit ;
    -> end//
       Query OK, 0 rows affected (0.00 sec)
mysql> delimiter ;
```

调用存储过程 sp_cno()，运行结果如下：

```
mysql> call sp_cno();
       +---------+--------+-----+------+----+-----+
       | courseno | cname   | type | period | exp | term |
       +---------+--------+-----+------+----+-----+
       | c05139   | 建模UML | 选修 |   48 | 12 |  7  |
```

```
+---------+-------+-----+------+----+-----+
1 row in set (0.06 sec)
Query OK, 0 rows affected (0.08 sec)
```


本例定义了一个事务,向表 course 添加一条记录,并设置保存点 spcno1。删除该记录之后,回滚到事务的保存点 spcno1 处,使用 commit 提交事务。最终的结果是记录没有被删除。

【例 9-5】 编写转账业务的存储过程,要求 bank 表中的账户的当前金额 cur_money 值不能小于 1。

代码和运行结果如下:

```
mysql> use mysqltest;
       Database changed
# 创建表 bank,输入记录并显示
mysql> create table bank(
    -> cus_no varchar(8),
    -> cus_name varchar(10),
    -> cur_money decimal(13,2));
       Query OK, 0 rows affected (0.64 sec)
mysql> insert into bank values('bj101211','张思睿',1000);
       Query OK, 1 row affected (0.04 sec)
mysql> insert into bank values('sd101677','李佛',1);
       Query OK, 1 row affected (0.03 sec)
mysql> select * from bank ;
       +--------+--------+--------+
       | cus_no   | cus_name | cur_money |
       +--------+--------+--------+
       | bj101211 | 张思睿   | 1000.00  |
       | sd101677 | 李佛     |    1.00  |
       +--------+--------+--------+
       2 rows in set (0.01 sec)
# 创建存储过程 trans_bank()
mysql> delimiter //
mysql> Create procedure trans_bank()
    -> begin
    -> declare money decimal(13,2);
    -> start transaction;
    -> update bank set cur_money = cur_money - 1000 where cus_no = 'bj101211';
    -> update bank set cur_money = cur_money + 1000 where cus_no = 'sd101677';
    -> select cur_money into money from bank where cus_no = 'bj101211';
    -> if money < 1 then
    -> begin
    -> select 'The transaction fails, the rollback transaction ';
    -> rollback;
    -> end;
    -> else
    -> begin
    -> select 'A successful transaction,commits the transaction';
```

```
    -> commit;
    -> end;
    -> end if;
    -> end//
       Query OK, 0 rows affected (0.00 sec)
mysql> delimiter ;
```

调用存储过程 trans_bank(),查看事务的执行结果如下：

```
mysql> call trans_bank();
        +----------------------------------------+
        | The transaction fails, the rollback transaction  |
        +----------------------------------------+
        | The transaction fails, the rollback transaction  |
        +----------------------------------------+
        1 row in set (0.04 sec)
        Query OK, 0 rows affected (0.06 sec)
```

9.3 事务的并发处理

用户创建会话访问服务器时,系统会为用户分配私有内存区域,保存当前用户的数据和控制信息,每个用户进程通过访问自己的私有内存区访问服务器,用户之间互不干扰,以此实现并发数据访问的控制。当数据库引擎所支持的并发操作数较大时,数据库并发程序就会增多。控制多个用户如何同时访问和更改共享数据而不会彼此冲突称为并发控制。

在 MySQL 中,并发控制是通过锁来实现的。如果事务与事务之间存在并发操作,事务的隔离性是通过事务的隔离级别来实现的,而事务的隔离级别则是由事务并发处理的锁机制来管理的。以此保证同一时刻执行多个事务时,一个事务的执行不能被其他事务干扰。

9.3.1 并发问题及其影响

多个用户访问同一个数据资源时,如果数据存储系统没有并发控制,就会出现并发问题,比如修改数据的用户会影响同时读取或修改相同数据的其他用户。当同一数据库系统中有多个事务并发运行时,如果不加以适当控制,就可能产生数据的不一致性问题。

下面以并发取款操作为例,介绍并发操作过程中的常见问题。如果得到错误的结果往往是由于 T1、T2 两个事务并发操作引起的,数据库的并发操作导致数据库的不一致性主要有 4 种：更新丢失、不可重复读、“脏读”和幻读数据。另外,数据库的并发操作还能够导致死锁问题发生。

(1) 更新丢失(Lost Update)：当两个或多个事务选择同一行,然后根据最初选定的值更新该行时,就会出现更新丢失的问题。每个事务都不知道其他事务的存在。最后的更新将覆盖其他事务所做的更新,从而导致数据丢失。

如表 9-1 所示,假设某客户存款的金额 M=2000 元,事务 T1 取走存款 500 元,事务 T2 取走存款 800 元,如果正常操作,即甲事务 T1 执行完毕再执行乙事务 T2,存款金额更新后应该是 700 元。但是如果按照如下顺序操作,则会有不同的结果。

表 9-1 更新丢失

时间	事务 T1	M 的值	事务 T2
t0		2000	
t1	select M		
t2			select M
t3	M=M-500		
t4			M=M-800
t5	update M		
t6		1500	update M
t7		1200	

① T1 事务开始读取存款金额 M=2000 元。

② T2 事务开始读取存款金额 M=2000 元。

③ T1 事务取走存款 500 元,修改存款金额 M=M-500=1500,把 M=1500 写回到数据库。

④ T2 事务取走存款 800 元,修改存款金额 M=M-800=1200,把 M=1200 写回到数据库。

结果两个事务共取走存款 1300 元,而数据库中的存款却只少了 800 元。

(2) 脏读(Dirty Read):即读出的是不正确的临时数据。例如,T2 事务选择 T1 事务正在更新的行时,就会出现一个事务可以读到另一个事务未提交的数据。T2 事务正在读取的数据尚未被 T1 事务提交,并可能由更新此行 T1 事务更改。脏读问题违背了事务的隔离性原则。

如表 9-2 所示,T2 事务读取的数据 1500,是尚未被 T1 事务提交的数据,回滚操作后,金额 M 仍然是 2000,而 T2 事务却读出 1500。

表 9-2 脏读

时间	事务 T1	M 的值	事务 T2
t0		2000	
t1	select M		
t2	M=M-500		
t3	Update M		
t4		1500	select M
t5	rollback		
t6		2000	

(3) 不可重复读(Non-repeatable Reads):同一个事务内两条相同的查询语句,查询结果不一致。即当一个事务多次访问同一行且每次读取不同数据时,会出现不可重复读问题。因为其他事务可能正在更新该事务正在读取的数据。如表 9-3 所示,事务 T1 多次查询 M 值,得到不同的结果,原因是事务 T2 修改了数据。

表 9-3　不可重复读

时间	事务 T1	M 的值	事务 T2
t0		2000	
t1	Select M		
t2			Select M
t3			M=M-800
t4			Update M
t5	Select M	1200	

(4) 幻读(Phantom Read)：当对某行执行插入或删除操作，而该行属于某事务正在读取的行的范围时，就会出现幻读问题。由于其他事务的删除操作，使事务第一次读取行范围时存在的行在后续读取时已不存在。与此类似，由于其他事务的插入操作，后续读取显示原来读取时并不存在的行。例如，T1 事务第一次执行查询操作，查看 M 值为 2000。T2 事务删除本行记录后，T1 事务第二次执行查询操作，查看 M 值，记录为 NULL。T2 事务插入该行记录后，T1 事务第三次执行查询操作，查看 M 值为 2000，如表 9-4 所示。

表 9-4　幻读

时间	事务 T1	M 的值	事务 T2
t0		2000	
t1	Select M		
t2			Delete M
t3	Select M	NULL	
t4			Insert M
t5	Select M	2000	

(5) 死锁(Deadlock)：如果很多用户并发访问数据库的话，还有一个常见的现象就是死锁。简单地说，如果两个用户相互等待对方的数据，就产生了一个死锁。mysql 检测到死锁之后，会选择一个事务进行回滚。而选择的依据为：看哪个事务的权重最小，事务权重的计算方法：事务加的锁最少；事务写的日志最少；事务开启的时间最晚。例如，事务 T2 写了日志，事务 T1 没有，回滚事务 T1。事务 T1、T2 都没写日志，但是事务 T1 开始得早，回滚事务 T2。

9.3.2　设置事务的隔离级别

为了防止数据库的并发操作导致的数据库不一致性的更新丢失、不可重复读、“脏读”和幻读数据等问题。SQL 标准定义了 4 种隔离级别：read uncommitted(读取未提交的数据)、read committed(读取提交的数据)、repeatable read(可重复读)以及 serializable(串行化)。4 种隔离级别逐渐增强，其中 read uncommitted 的隔离级别最低，serializable 的隔离级别最高。

MySQL 支持 4 种事务隔离级别，在 InnoDB 存储引擎中，可以使用以下命令设置事务的隔离级别。

```
set {global|session}transaction isolation level{
```

```
read uncommitted|read committed|repeatable read|serializable}
```

说明：

(1) read uncommitted：在该隔离级别，所有事务都可以看到其他未提交事务的执行结果。该隔离级别很少用于实际应用，并且它的性能也不比其他隔离级别好多少。

(2) read committed：这是大多数数据库系统的默认隔离级别(但不是 MySQL 默认的)。它满足了隔离的简单定义，即一个事务只能看见已提交事务所做的改变。

(3) repeatable read：这是 MySQL 默认的事务隔离级别，它确保同一事务内相同的查询语句执行结果一致。

(4) serializable：这是最高的隔离级别，它通过强制事务排序，使之不可能相互冲突。换言之，它会在每条 select 语句后自动加上 lock in share mode，为每个查询操作施加一个共享锁。在这个级别，可能导致大量的锁等待现象。该隔离级别主要用于 InnoDB 存储引擎的分布式事务。

低级别的事务隔离可以提高事务的并发访问性能，却可能导致较多的并发问题(例如脏读、不可重复读、幻读等并发问题)；高级别的事务隔离可以有效避免并发问题，但会降低事务的并发访问性能，可能导致出现大量的锁等待、甚至死锁现象。

系统变量@@tx_isolation 中存储了事务的隔离级别，可以使用 select 随时获得当前隔离级的值，如下所示：

```
mysql> select @@tx_isolation
```

MySQL 默认为 repeatable read 隔离级，这个隔离级适用于大多数应用程序，只有在应用程序有具体的对于更高或更低隔离级的要求时才需要改动。没有一个标准公式来决定哪个隔离级适用于应用程序，一般是基于应用程序的容错能力和应用程序开发者对于潜在数据错误的影响的经验判断。隔离级的选择对于每个应用程序也是没有标准的。

9.4 管 理 锁

多用户同时并发访问同一数据表时，仅仅通过事务机制，是无法保证数据的一致性的，MySQL 通过锁来防止数据并发操作过程中引起的问题。锁就是防止其他事务访问指定资源的手段，它是实现并发控制的主要方法，是多个用户能够同时操作同一个数据库中的数据而不发生数据不一致性现象的重要保障。

9.4.1 认识锁机制

MySQL 引入锁机制管理的并发访问，通过不同类型的锁来管理多用户并发访问，实现数据访问的一致性。

1. 锁机制中的基本概念

(1) 锁的粒度。锁的粒度是指锁的作用范围。锁的粒度可以分为服务器级锁(server-level locking)和存储引擎级锁(storage-engine-level locking)。MyISAM 存储引擎支持表锁。InnoDB 存储引擎支持表锁以及行级锁。

(2) 隐式锁与显式锁。MySQL 锁分为隐式锁以及显式锁。MySQL 自动加锁称为隐

式锁。数据库开发人员手动加锁称为显式锁。

(3) 锁的类型。锁的类型包括读锁(read lock)和写锁(write lock),其中读锁也称为共享锁,写锁也称为排他锁或者独占锁。读锁允许其他 MySQL 客户机对数据同时"读",但不允许其他 MySQL 客户机对数据任何"写"。写锁不允许其他 MySQL 客户机对数据同时读,也不允许其他 MySQL 客户机对数据同时写。

(4) 锁的钥匙。多个 MySQL 客户机并发访问同一个数据时,如果 MySQL 客户机 A 对该数据成功地施加了锁,那么只有 MySQL 客户机 A 拥有这把锁的"钥匙",也就是说,只有 MySQL 客户机 A 能够对该锁进行解锁操作。

(5) 锁的生命周期。锁的生命周期是指在同一个 MySQL 服务器连接内,对数据加锁到解锁之间的时间间隔。

2. 锁定与解锁

(1) 锁定表。MySQL 提供了 lock tables 语句来锁定当前线程的表。

锁定表的语法格式如下:

```
lock tables table_name[as alias]{read[local]|[low_priority]write}
```

说明:

表锁定支持以下类型的锁定。read 锁确保用户可以读取表,但是不能修改表。write 锁只有锁定该表的用户可以修改表,其他用户无法访问该表。

在对一个事务表使用表锁定的时候需要注意的是:在锁定表时会隐式地提交所有事务,在开始一个事务时,如 start transaction,会隐式解开所有表锁定。在事务表中,系统变量@@autocommit 值必须设为 0。否则,MySQL 会在调用 lock tables 之后立刻释放表锁定,并且很容易形成死锁。

例如,在 score 表上设置一个只读锁定:

```
lock tables score read;
```

在 course 表上设置一个写锁定:

```
lock tables course write;
```

(2) 解锁表。在锁定表以后,可以使用 unlock tables 命令解除锁定。该命令不需要指出解除锁定的表的名字。

解锁表的语法格式为如下:

```
unlock tables;
```

9.4.2 锁的分类

MySQL 支持很多不同的表类型,而且对于不同的类型,锁定机制也是不同的。在 MySQL 中有 3 种锁定机制。

(1) 表锁:一个特殊类型的访问,整个表被客户锁定。根据锁定的类型,其他客户不能向表中插入记录,甚至从中读数据也受到限制。表级锁定的类型包括两种锁,即读锁(read)和写锁(write)。

- 读锁。用于表级锁定的实现机制如下：如果表没有加写锁，那么就加一个读锁。否则的话，将请求放到读锁队列中。
- 写锁。用于表级锁定的实现机制如下：如果表没有加锁，那么就加一个写锁。否则的话，将请求放到写锁队列中。

（2）行锁：行级的锁定比表级锁定或页级锁定对锁定过程提供了更精细的控制。在这种情况下，只有线程使用的行是被锁定的。表中的其他行对于其他线程都是可用的。行级锁定并不是由 MySQL 提供的锁定机制，而是由存储引擎自己实现的，其中 InnoDB 的锁定机制就是行级锁定。行级锁定的类型包括 3 种：排他锁、共享锁和意向锁（参见表 9-5）。

表 9-5　请求锁模式的兼容性

	X	IX	S	IS
排他锁（X）	冲突	冲突	冲突	冲突
意向排他锁（IX）	冲突	兼容	冲突	兼容
共享锁（S）	冲突	冲突	兼容	兼容
意向共享锁（IS）	冲突	兼容	兼容	兼容

- 排他锁（eXclusive Locks）。排他锁又称为 X 锁。如果事务 T1 获得了数据行 D 上的排他锁，则 T1 对数据行既可读又可写。事务 T1 对数据行 D 加上排他锁，则其他事务对数据行 D 的任务封锁请求都不会成功，直至事务 T1 释放数据行 D 上的排他锁。
- 共享锁（Share Locks）。共享锁又称为 S 锁。如果事务 T1 获得了数据行 D 上的共享锁，则 T1 对数据项 D 可以读但不可以写。事务 T1 对数据行 D 加上共享锁，则其他事务对数据行 D 的排他锁请求不会成功，而对数据行 D 的共享锁请求可以成功。
- 意向锁。意向锁是一种表锁，锁定的粒度是整张表，分为意向共享锁（IS）和意向排他锁（IX）两类。意向锁表示一个事务有意对数据上共享锁或排他锁。
  - ★ 意向共享锁（IS）：事务打算给数据行加共享锁，事务在取得一个数据行的共享锁之前必须先取得该表的 IS 锁。
  - ★ 意向排他锁（IX）：事务打算给数据行加排他锁，事务在取得一个数据行的排他锁之前必须先取得该表的 IX 锁。

InnoDB 表的行锁是通过对“索引”施加锁的方式实现的，这就是说，只有通过索引字段检索数据的查询语句或者更新语句，才可能施加行级锁；否则 InnoDB 将使用表级锁，使用表级锁势必会降低 InnoDB 表的并发访问性能。

（3）页锁：MySQL 将锁定表中的某些行（称作页）。被锁定的行只对锁定最初的线程是可行的。

BDB 表支持页级锁。页级锁开锁和加锁时间介于表级锁和行级锁之间，会出现死锁，锁定粒度介于表级锁和行级锁之间。

InnoDB 表速度很快，比 BDB 更有性能优势，InnoDB 表适合执行大量的 insert 或 update 数据操作。影响 InnoDB 类型的表速度的主要原因是 autocommit 默认设置是打开的，如果程序没有显式调用 begin 开始事务，会导致每插入一条数据都会自动提交，严重影响了速度。

在查询(select)语句或者修改(insert、update 以及 delete)语句中,为受影响的记录施加行级锁的方法也非常简单。在修改数据时,InnoDB 存储引擎将符合更新条件的记录自动施加排他锁(隐式锁)。即 InnoDB 存储引擎自动地为更新语句影响的记录施加隐式排他锁。例如,在 select 语句中,为符合查询条件的记录施加共享锁示例和为符合查询条件的记录施加排他锁示例。

```
select * from student where sex = '女' lock in share mode;
select * from student where entrance > = 850 for update;
```

9.4.3 死锁的管理

1. 死锁的原因

两个或两个以上的事务分别申请封锁对方已经封锁的数据对象,导致长期等待而无法继续运行下去的现象称为死锁。

MySQL 对并发事务的处理,使用任何方案都会导致死锁问题。在下面两种情况会经常发生死锁现象。

第 1 种情况是,两个事务分别锁定了两个单独的对象,这时每一个事务都要求在另外一个事务锁定的对象上获得一个锁,结果是每一个事务都必须等待另外一个事务释放占有的锁,此时就发生了死锁。这种死锁是最典型的死锁形式。

第 2 种情况是,在一个数据库中,有若干长时间运行的事务并行的执行操作,查询分析器处理非常复杂的查询时,例如连接查询,由于不能控制处理的顺序,有可能发生死锁。

死锁是指事务永远不会释放它们所占用的锁,死锁中的两个事务都将无限期等待下去。MySQL 的 InnoDB Engine 自动检测死锁循环,并选择一个会话作为死锁中放弃的一方,通过终止该事务来打断死锁。被终止的事务发生回滚,并返回给连接一个错误消息。

如果在交互式的 MySQL 语句中发生死锁错误,用户只要简单地重新输入该语句即可。

2. 死锁的处理

默认情况下,InnoDB 存储引擎一旦出现锁等待超时异常,InnoDB 存储引擎既不会提交事务,也不会回滚事务,而这是十分危险的。一旦发生锁等待超时异常,应用程序应该自定义错误处理程序,由程序开发人员选择进一步提交事务,还是回滚事务。

在 InnoDB 的事务管理和锁定机制中,有专门用于检测死锁的机制。当检测到死锁时,InnoDB 会选择产生死锁的两个事务中较小的一个产生回滚,而让另外一个较大的事务成功完成。那么如何判断事务的大小呢?主要是通过计算两个事务各自插入、更新或者删除的数据量来判断,也就是说哪个事务改变的记录数越多,在死锁中越不会被回滚。需要注意的是,如果在产生死锁的场景中涉及不止 InnoDB 存储引擎时,InnoDB 是检测不到该死锁的,这时就只能通过锁定超时限制来解决该死锁了。

3. 事务与锁机制注意事项

(1) 锁的粒度越小,应用系统的并发性能就越高,由于 InnoDB 存储引擎支持行锁,建议使用 InnoDB 存储引擎表以提高系统的可靠性。

(2) 使用事务时,尽量避免一个事务中使用不同存储引擎的表。

(3) 处理事务时尽量设置和使用较低的隔离级别。

(4) 尽量使用基于行锁控制的隔离级别。必要时使用表锁,可以避免死锁现象。

(5) 对于 InnoDB 存储引擎支持的行锁,设置合理的超时参数范围,编写锁等待超时异常处理程序,可以解决锁等待问题。

(6) 为避免死锁,事务进行多记录修改时,尽量在获得所有记录的排他锁后,再进行修改操作。

(7) 为避免死锁,尽量缩短锁的生命周期,保持事务简短并处于一个批处理中。

(8) 为避免死锁,事务中尽量按照同一顺序访问数据库对象,避免在事务中存在用户交互访问数据的情况。

9.5 小　结

通过本章的学习,读者需要认识到 MySQL 中所有的数据访问都是通过事务进行的,了解 MySQL 如何在事务间通过锁来实现并发控制。事务的 ACID 特性及锁的使用,目的都是为了保证数据的一致性和完整性。通过学习要求掌握如下的内容。

- 事务和锁的基本概念。
- 事务的定义、启动和应用场合。
- 如何通过定义隔离级别实现事务访问资源和数据的隔离,以及隔离级别与并发问题的关系。
- 锁的类型和管理。

习　题　9

1. 选择题

(1) MySQL 的事务不具有的特征是________。

A. 原子性　　B. 隔离性　　C. 一致性　　D. 共享性

(2) MySQL 中常见的锁类型不包括________。

A. 共享　　B. 意向　　C. 架构　　D. 排他

(3) 事务的隔离级别不包括________。

A. read uncommitted　　B. read committed

C. repeatable read　　D. repeatable only

(4) 死锁发生的原因是________。

A. 并发控制　　B. 服务器故障　　C. 数据错误　　D. 操作失误

(5) MySQL 中发生死锁需要________。

A. 用户处理　　B. 系统自动处理　　C. 修改数据源　　D. 取消事务

2. 思考题

(1) 简述并发控制可能产生的影响,分别描述产生的原因。

(2) 如何设置事务的隔离级别?

(3) 如何在事务中设置保存点,保存点有什么用途?

(4) 什么是死锁,哪些方法可以解除死锁?

（5）简述 MySQL 中锁的粒度及锁的常见类型。

3. 上机练习题（本题利用 teaching 数据库中的表进行操作）

（1）创建存储过程 up_score，实现在 score 表上执行 update 语句的事务，并执行存储过程进行验证。

（2）定义一个事务，向 course 表中添加一条记录，并设置保存点。然后再删除该记录，并回滚到事务的保存点，提交事务。

（3）创建事务，练习在 student 表上进行查询、插入和更新操作。

（4）将 course 表中课程号为 c08123 的课程名称改为“PHP 语言”，并提交该事务。

第 10 章　权限管理及安全控制

数据库的安全性是指保护数据库以防止不合法使用所造成的数据泄露、更改或破坏。系统安全保护措施是否有效是数据库系统主要的性能指标之一。数据库的安全性与计算机系统的安全性紧密联系，数据库管理系统提供的主要技术有强制存取控制、数据加密存储和加密传输等。控制数据存取流程，通过用户标识和鉴定、存取控制、视图、审计和数据加密等方法，将非法用户和不具备完整性的数据进行特别处理。

在 MySQL 数据库管理系统中，主要是通过用户权限管理实现其安全性控制的。当在服务器上运行 MySQL 时，数据库管理员的职责就是要想方设法使 MySQL 免遭用户的非法侵入，拒绝其访问数据库，保证数据库的安全性和完整性。

10.1　MySQL 权限系统的工作原理

了解 MySQL 数据库的安全性，首先要了解 MySQL 的访问控制系统，掌握 MySQL 权限系统的工作原理，熟悉其权限操作。当 MySQL 服务启动时，首先会读取 MySQL 中的权限表，并将表中的数据装入内存。当用户进行存取操作时，MySQL 会根据这些表中的数据做相应的权限控制。

10.1.1　MySQL 的权限表

通过网络连接服务器的客户对 MySQL 数据库的访问由权限表内容来控制。用户登录以后，MySQL 数据库系统会根据这些权限表的内容为每个用户赋予相应的权限。这些权限表中最重要的是 user 表、db 表和 host 表。除此之外，还有 tables_priv 表、columns_priv 表、proc_priv 表等。

1. user 表

user 表是 MySQL 中最重要的一个权限表，记录允许连接到服务器的账号信息。user 表列出可以连接服务器的用户及其口令，并且指定他们有哪种全局(超级用户)权限。在 user 表启用的任何权限均是全局权限，并适用于所有数据库。MySQL 5.7 中 user 表有 45 个字段，这些字段共分为 4 类，分别是用户列、权限列、安全列和资源控制列。利用“mysql>desc user;”可以查看 user 表结构。运行结果(部分)显示如下：

```
mysql> use mysql;
       Database changed
mysql> desc user;
```

```
+-----------+-----------+----+----+-------+----+
| Field     | Type      | Null | Key | Default |Extra|
+-----------+-----------+----+----+-------+----+
| Host      | char(60)  | NO | PRI |       |    |
| User      | char(32)  | NO | PRI |       |    |
| Select_priv | enum('N','Y') | NO |    | N     |    |
| Insert_priv | enum('N','Y') | NO |    | N     |    |
| Update_priv | enum('N','Y') | NO |    | N     |    |
| Delete_priv | enum('N','Y') | NO |    | N     |    |
| Create_priv | enum('N','Y') | NO |    | N     |    |
| Drop_priv   | enum('N','Y') | NO |    | N     |    |
...
+-----------+-----------+----+----+-------+----+
45 rows in set (0.18 sec)
```

从结果中,可以看到用户的常见权限字段定义。例如,如果用户获得了 delete 权限,就可以从表中删除记录。其他权限表也可以采用同样的方式来查看。

2. db 表和 host 表

db 表和 host 表也是 MySQL 数据库中非常重要的权限表。db 表中存储了用户对某个数据库的操作权限,决定用户能从哪个主机存取哪个数据库。host 表中存储了某个主机对数据库的操作权限,配合 db 权限表对给定主机上数据库级的操作权限做更细致的控制。这个权限表不受 grant 和 revoke 语句的影响。db 表比较常用,host 表一般很少使用。db 表和 host 表的字段大致可以分为两类,分别是用户列和权限列。

3. tables_priv 表和 columns_priv 表

tables_priv 表可以对单个表进行权限设置,tables_priv 表包含 8 个字段,分别是 Host、Db、User、Table_name、Table_priv、Column_priv、Timestamp 和 Grantor。前 4 个字段分别表示主机名、数据库名、用户名和表名。Table_priv 表示对表进行操作的权限。这些权限包括 Select、Insert、Update、Delete、Create、Drop、Grant、References、Index 和 Alter。Columns_priv 表示对列操作权限。Timestamp 表示修改权限的时间。Grantor 表示权限是谁设置的。

columns_priv 表可以对单个数据列进行权限设置,包含 7 个字段。Columns_priv 表示对表中的数据列进行操作的权限。这些权限包括 Select、Insert、Update 和 References。

4. procs_priv 表

procs_priv 表可以存储过程和存储函数进行权限设置。procs_priv 表包含 8 个字段,分别是 Host、Db、User、Routine_name、Routine_type、Proc_priv、Timestamp 和 Grantor。前三个字段分别表示主机名、数据库名和用户名。Routine_name 字段表示存储过程或函数的名称。Routine_type 字段表示类型。该字段有两个取值,分别是 function 和 procedure。function 表示这是一个存储函数。procedure 表示这是一个存储过程。Proc_priv 字段表示拥有的权限。

权限分为 3 类,分别是 Execute、Alter Routine 和 Grant。Timestamp 字段存储更新的时间。Grantor 字段存储权限是谁设置的。

10.1.2 MySQL 权限系统的工作过程

为了确保数据库的安全性与完整性,数据库系统并不希望每个用户可以执行所有的数

据库操作。当 MySQL 允许一个用户执行各种操作时，将首先核实用户向 MySQL 服务器发送的连接请求，然后确认用户的操作请求是否被允许。MySQL 的访问控制分为两个阶段：连接核实阶段和请求核实阶段。

1. 连接核实阶段

当用户试图连接 MySQL 服务器时，服务器基于用户提供的信息来验证用户身份，如果不能通过身份验证，服务器会完全拒绝该用户的访问。如果能够通过身份验证，则服务器接受连接，然后进入第 2 个阶段等待用户请求。

MySQL 使用 user 表中的 3 个字段(Host、User 和 Password)进行身份检查，服务器只有在用户提供主机名、用户名和密码并与 user 表中对应的字段值完全匹配时才接受连接。

2. 请求核实阶段

一旦连接得到许可，服务器进入请求核实阶段。在这一阶段，MySQL 服务器对当前用户的每个操作都进行权限检查，判断用户是否有足够的权限来执行它。用户的权限保存在 user、db、host、tables_priv 或 columns_priv 权限表中。

在 MySQL 权限表的结构中，user 表在最顶层，是全局级的。下面是 db 表和 host 表，它们是数据库层级的。最后才是 tables_priv 表和 columns_priv 表，它们是表级和列级的。低等级的表只能从高等级的表得到必要的范围或权限。

如图 10-1 所示，MySQL 接收到用户的操作请求时，首先确认用户是否有权限。如果没有，则 MySQL 首先检查 user 表，即先检查全局权限表 user，如果 user 中对应的权限为 T(有)，则此用户对所有数据库的权限为 T，将不再检查 db、tables_priv、columns_priv；如果为 F(无)，则从 db 表中检查此用户对应的具体数据库，并得到 db 中的 T 的权限；如果 db 中为 F，则检查 tables_priv 及 columns_priv 表中此数据库对应的具体表，取得表中的权限 T，以此类推。如果所有权限表都检查完毕，依旧没有找到允许的权限操作，MySQL 服务器将返回错误信息，用户操作不能执行，操作失败。

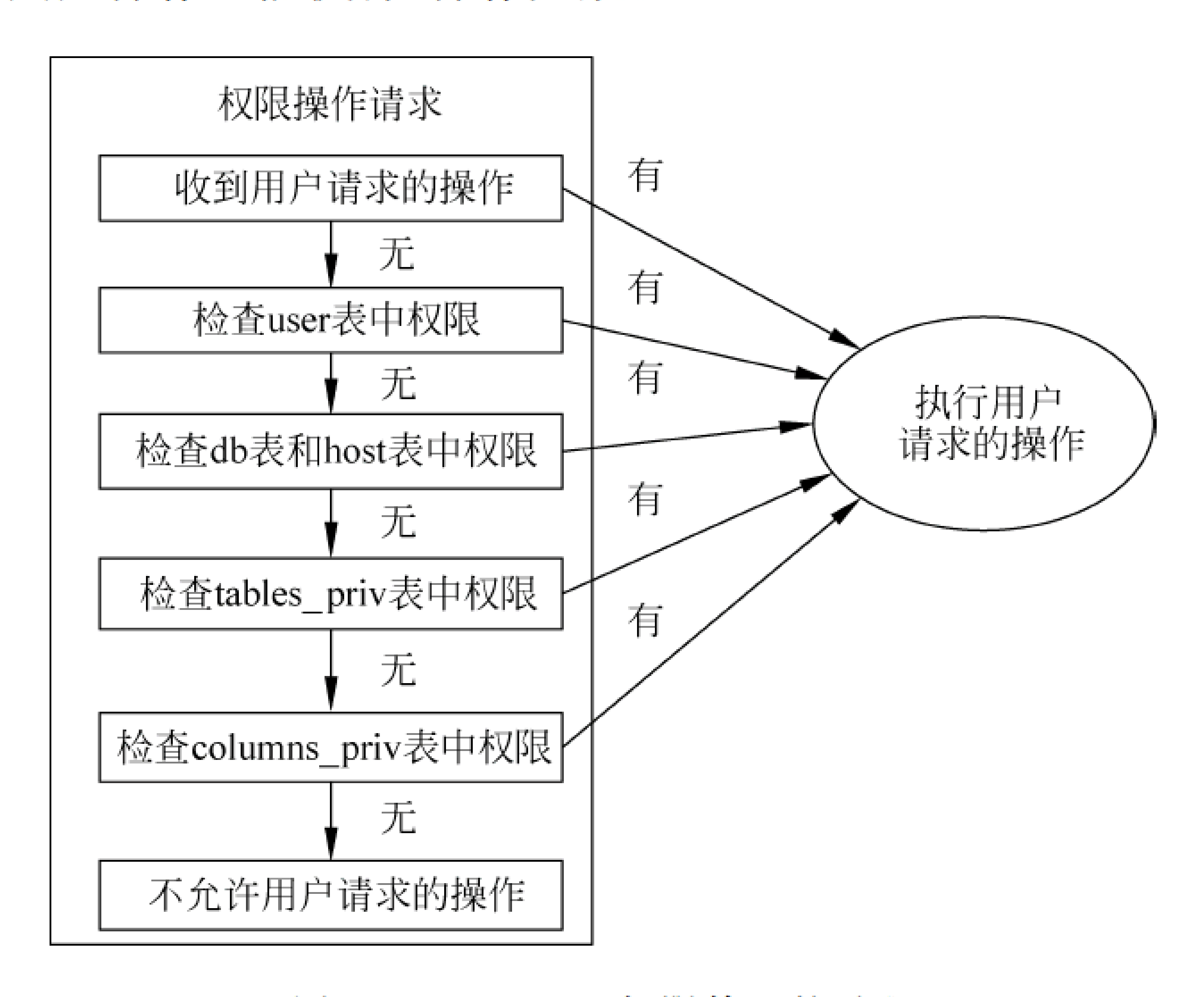


图 10-1　MySQL 权限管理的过程

10.2 账户管理

账户管理是 MySQL 用户管理的最基本的内容。账户管理包括登录和退出 MySQL 服务器、创建用户、删除用户、密码管理、权限管理等内容。通过账户管理，可以保证 MySQL 数据库的安全性。

10.2.1 普通用户的管理

MySQL 用户包括普通用户和 root 用户。这两种用户的权限是不一样的。root 用户是超级管理员，拥有所有的权限。root 用户的权限包括创建用户、删除用户、修改普通用户的密码等管理权限。而普通用户只拥有创建该用户时赋予它的权限。用户管理包括管理用户的账户、权限等。

1. 使用 create user 语句创建新用户



创建用户

在 MySQL 数据库中，创建新用户可以直接操作 MySQL 权限表，也可以使用 create user 语句或 grant 语句。要使用 create user 语句，必须拥有 MySQL 数据库的全局 create user 权限，或拥有 insert 权限。对于每个账户，create user 会在没有权限的 mysql. user 表中创建一个新记录。如果账户已经存在，则出现错误。使用自选的 identified by 子句，可以为账户设置一个密码。user 值和密码的设置方法与 grant 语句一样。

执行 create user 或 grant 语句时，服务器会有相应的用户权限表，添加或修改用户及其权限。

create user 语句的基本语法格式如下：

```
create user user[identified by [password]'password']
    [,user[identified by [password]'password']][,…];
```

说明：

(1) user 的格式为：'user_name'@ 'host name'。

(2) host name 指定了用户创建的使用 MySQL 的连接来自的主机。如果一个用户名和主机名中包含特殊符号如"_"，或通配符如"%"，则需要用单引号将其括起。"%"表示一组主机。

(3) localhost 表示本地主机。

(4) identified by 指定用户密码，注意用户名和密码区别大小写。

(5) password()是对密码进行加密。

(6) 可以使用 create user 语句同时创建多个数据库用户，用户名之间用逗号分隔。

【例 10-1】 添加两个新用户，Hans 的密码为 hans131，Rose 的密码为 rose123。

代码和运行结果如下：

```
mysql> create user
    -> 'Hans'@'localhost' identified by 'hans131',
    -> 'Rose'@'localhost' identified by 'rose123';
       Query OK, 0 rows affected (0.19 sec)
```

【例 10-2】 添加一个新用户,用户名为 Pool,密码为 136792,不指定明文。

操作步骤、代码和运行结果如下:

① 使用 password()函数获取密码'136792'的散列值。

```
mysql > select password('136792');
        +-------------------------------------------+
        | password('136792')                        |
        +-------------------------------------------+
        | *BB3D238C77F04751017773C8CFFFB291BFF7427C |
        +-------------------------------------------+
        1 row in set, 1 warning (0.00 sec)
```

② 执行 create user 语句创建用户 Pool。

```
mysql > create user 'Pool'@'localhost'
    -> identified by password'*BB3D238C77F04751017773C8CFFFB291BFF7427C';
       Query OK, 0 rows affected, 1 warning (0.00 sec)
```

2. 使用 grant 语句创建新用户

使用 grant 语句不仅可创建新用户,还可以在创建的同时对用户授权。grant 语句还可以指定用户的其他特点,如安全连接、限制使用服务器资源等。使用 grant 语句创建新用户时必须有 grant 权限。

grant 语句的基本语法格式如下:

```
grant priv_type on database.table
to user[identified by[password] 'password']
 [,user[identified by[password] 'password']][,...]
[with grant option];
```

【例 10-3】 使用 grant 语句创建一个新用户 test11,主机名为 localhost,密码为 test131,并授予所有数据表的 select 和 update 权限。

代码和运行结果如下:

```
mysql > grant select,update on *.* to 'test11'@'localhost'
    -> identified by 'test131';
       Query OK, 0 rows affected, 1 warning (0.01 sec)
```

其中,*.* 表示对用户 test131 设置全局权限。利用"select user from user;"语句可以验证新用户的添加是否成功。

3. 删除普通用户

如果存在一个或是多个账户被闲置,应当考虑将其删除,确保不会用于可能的违法的活动。利用 drop user 命令就能很容易地做到,它将从权限表中删除用户的所有信息,即来自所有授权表的账户权限记录。在 MySQL 数据库中,可以使用 drop user 语句来删除普通用户,也可以直接在 mysql. usr 表中删除用户。

(1) 使用 drop user 语句删除用户。

drop user 的语法格式如下:

```
drop user user_name[, user_name] [,...];
```

drop user 语句用于删除一个或多个 MySQL 账户，并取消其权限。要使用 drop user，必须拥有 MySQL 数据库的全局 create user 权限或 delete 权限。drop user 不能自动关闭任何打开的用户对话。如果用户有打开的对话，则取消用户，命令不会生效，直到用户对话被关闭后才生效。一旦对话被关闭，用户也被取消，此用户再次试图登录时将会失败。

例如，删除用户 TOM1 的命令如下：

```
mysql > drop user TOM1@localhost;
```

(2) 使用 delete 语句删除用户。delete 语句的基本语法格式如下：

```
delete from mysql.user where host = 'hostname' and user = 'username';
```

其中，host 和 user 为 user 表中的两个字段。

例如，使用 delete 删除用户 test11 的命令如下：

```
mysql > delete from mysql.user where host = 'localhost'and user = 'test11';
```

使用 select 语句查询 user 表中的记录，可以验证删除操作是否成功。

如果删除的用户已经创建了表、索引或其他的数据库对象，这些数据库对象将继续存在，因为 MySQL 并没有记录是谁创建了这些对象。

4. 修改用户名称

可以使用 rename user 语句来实现。如果旧账户不存在或者新账户已存在，则会出现错误。

使用 rename user 语句修改用户的基本语法格式如下：

```
rename user old_user to new_user,[,old_user to new_user] [,...];
```

例如，将用户 king11 和 king12 的名字分别修改为 king1 和 king1。可以使用如下命令：

```
mysql > rename user
    -> 'king11'@'localhost' to 'king1'@'localhost',
    -> 'king12'@'localhost' to 'king2'@'localhost';
```

10.2.2 MySQL 命令的使用

用户可以通过 MySQL 命令来登录 MySQL 服务器。前面已经简单介绍过一些登录 MySQL 服务器的方法，但是有些参数还不全。

1. 登录 MySQL 服务器

启动 MySQL 服务后，可以通过 MySQL 命令来登录 MySQL 服务器。

完整的 MySQL 命令如下：

```
mysql -h hostname|hostIP -P port -u username -p
Database Name -e "SQL statements";
```

说明：

(1) mysql：登录服务器的命令。

(2) -h hostname|hostIP：登录本地主机名|本地主机 IP 地址(127.0.0.1)。

(3) -p port：服务器的端口号，默认端口号为 3306，可省略。

(4) -u username：登录服务器的账户名，如 root。

(5) -p：用户名之后的参数-p 后是用户登录密码，一般是在输入该参数后按 Enter 键，然后直接输入登录密码。

(6) Database Name：要打开的数据库名。

(7) -e "SQL statements"：要执行的 SQL 语句字符串。

(8) 结束符：每条 SQL 语句都应该以“;”或“\g”结束。

2. 修改用户密码

要修改某个用户的登录密码，可以使用 mysqladmin 命令、update 语句或 set password 语句来实现。

(1) root 用户修改自己的密码。root 用户的安全对于保证 MySQL 的安全非常重要，因为 root 用户拥有全部权限。修改 root 用户密码的方式有多种。

① 使用 mysqladmin 命令。mysqladmin 命令的基础语法格式如下：

```
mysqladmin －u username －h localhost －p password "newpassword";
```

例如，可以使用 mysqladmin 命令将 root 用户的密码修改为"rootpwd"。

代码和运行结果如下：

```
mysql > mysqladmin －u root －p password "rootpwd";
Enter password: *******
```

② 使用 update 语句修改 MySQL 数据库中的 user 表。因为所有账户信息都保存在 user 表中，因此可以通过直接修改 user 表来改变 root 用户的密码。root 用户登录到 MySQL 服务器后，使用 update 语句修改 MySQL 数据库中的 user 表的 password 字段值，从而修改用户密码。

使用 update 语句修改 root 用户密码的语句如下：

```
mysql > update mysql.user set password = password('newpassword')
      where user = 'root' and host = 'localhost';
```

(2) 使用 set 语句修改用户密码。其基本语法格式如下：

```
set password [for user] =  password('newpassword');
```

例如，将用户 test11 的密码修改为 test12：

```
mysql > set password for 'test11'@'localhost' = password('test12');
```

10.2.3 利用图形工具管理用户

(1) 启动 MySQL Workbench 工具，单击实例 mysql57，进入初始界面。

(2) 在如图 10-2 所示的 MANAGEMENT(项目管理)中单击 Users and Privileges(用户和权限)选项，可以在如图 10-3 所示的 Users and Privileges 对话框中观察到已经创建的用户。

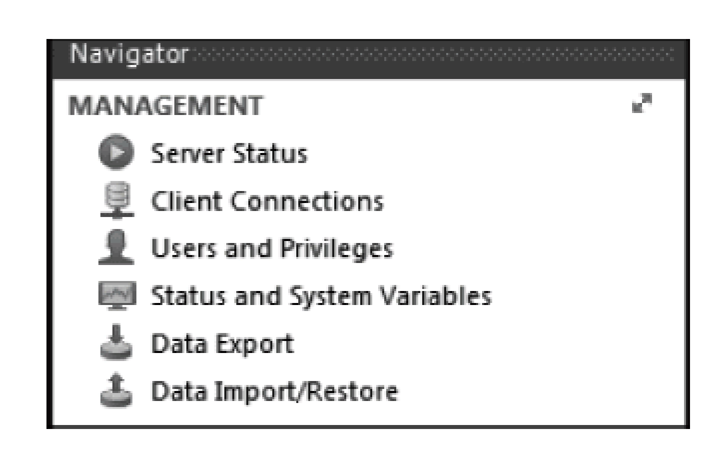


图 10-2 MySQL 的导航管理

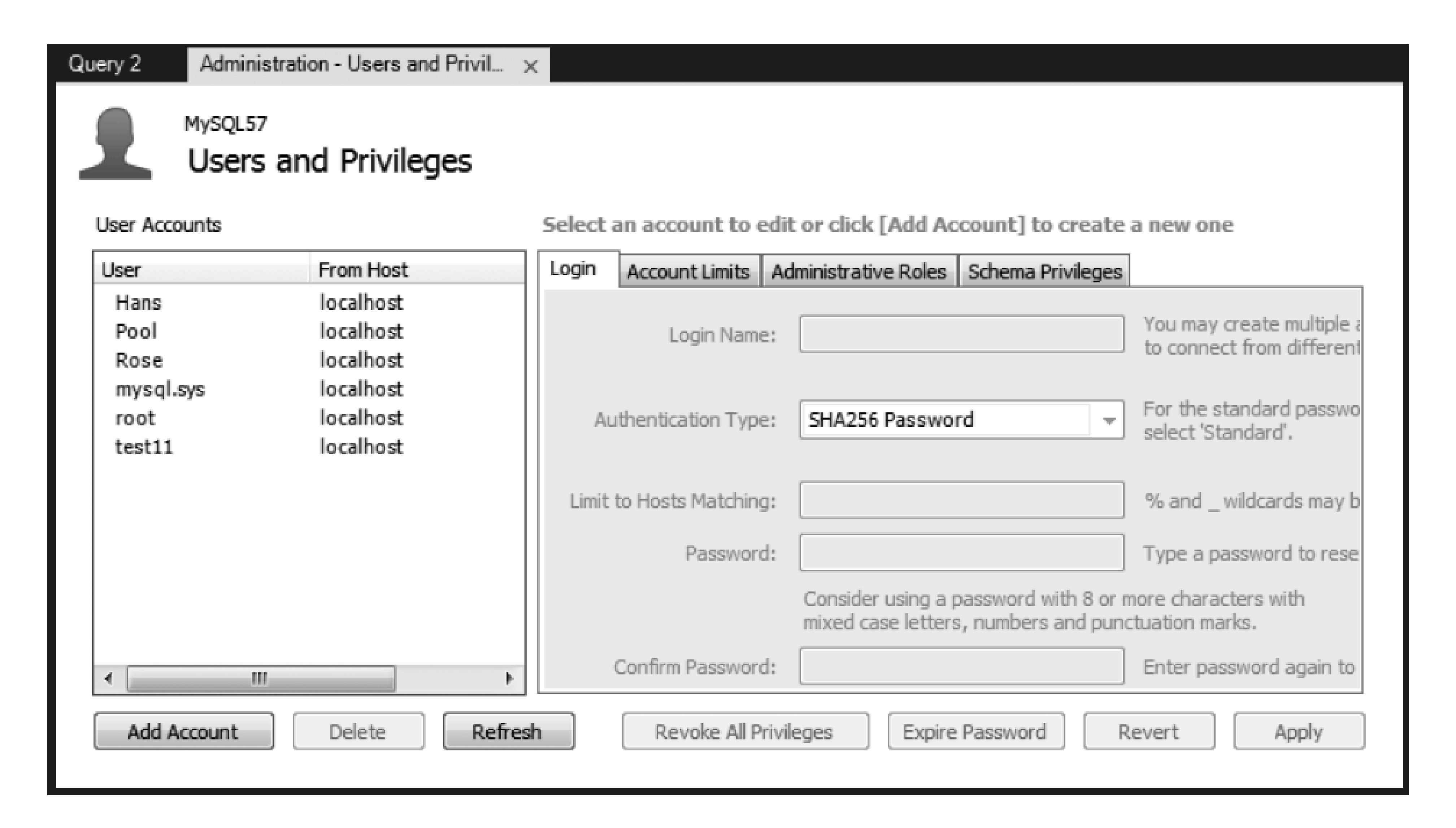


图 10-3　用户和权限管理

(3) 单击 Add Account(添加用户)按钮,就可以在如图 10-4 所示的界面中开始新建用户的过程。

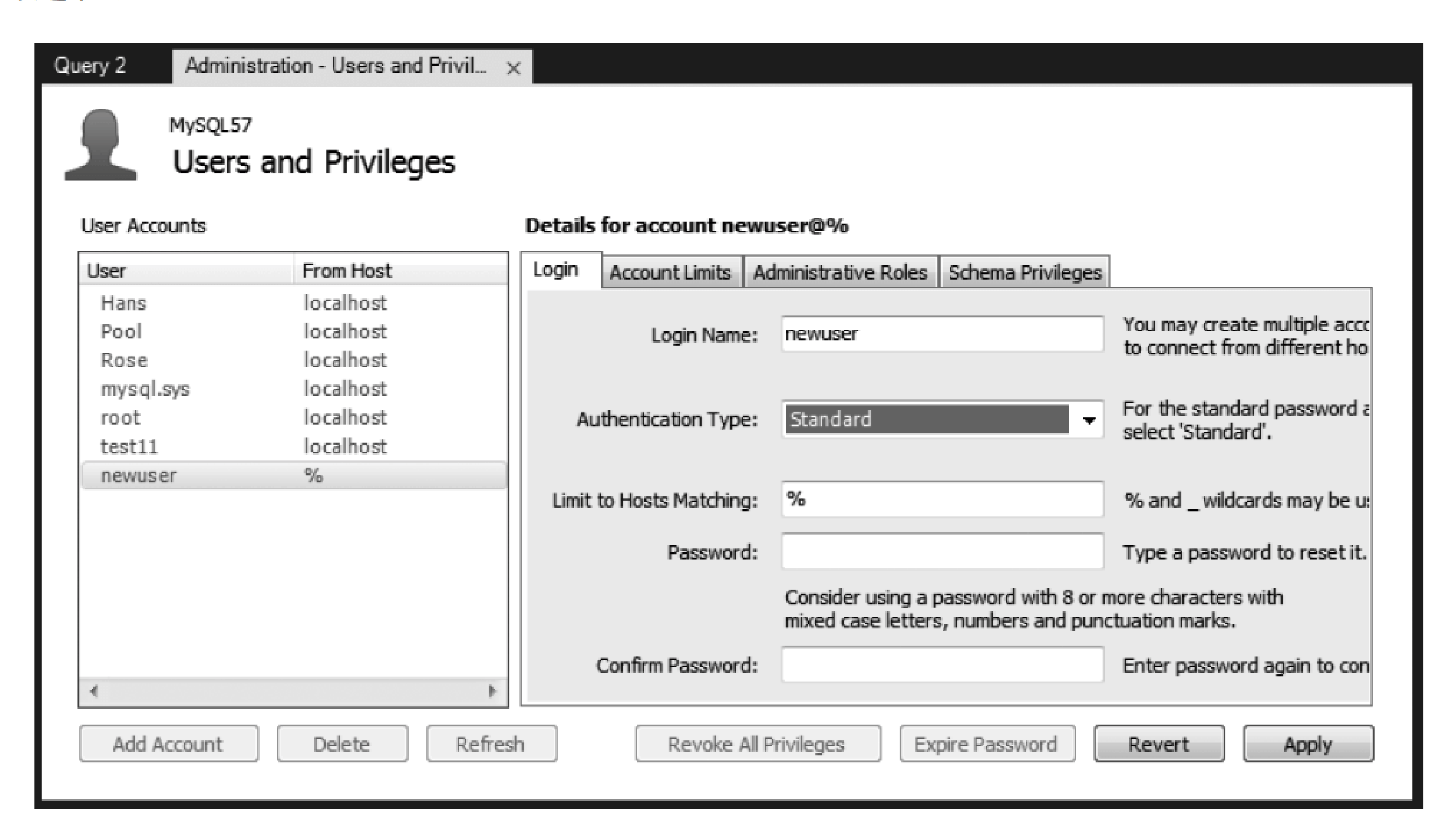


图 10-4　创建用户初始界面

(4) 如图 10-5 所示,输入用户名 test17,按指定要求输入密码如 abc123456。

(5) 如图 10-6 所示,进入数据库角色管理界面,选择适当的角色权限。打开数据库权限选项卡 Schema Privileges,单击 Add Entry 按钮,进入如图 10-7 所示的选择默认权限数据库界面,选择 mysqltest 数据库。

(6) 单击 OK 按钮,进入如图 10-8 所示的用户权限设置界面,为用户选择合适的权限,即可完成如图 10-9 所示的用户 test17 的创建过程。

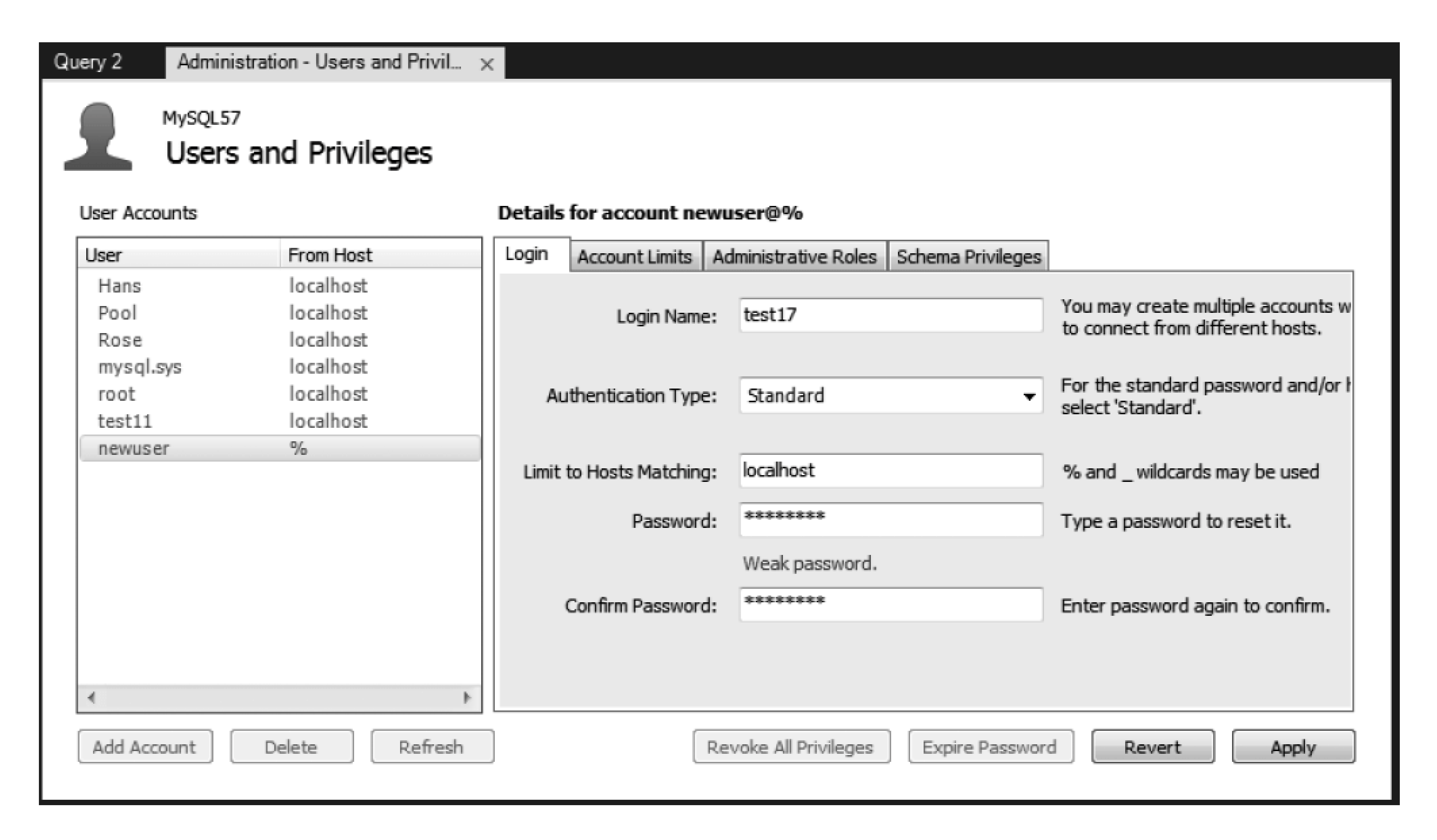


图 10-5　创建用户 test17

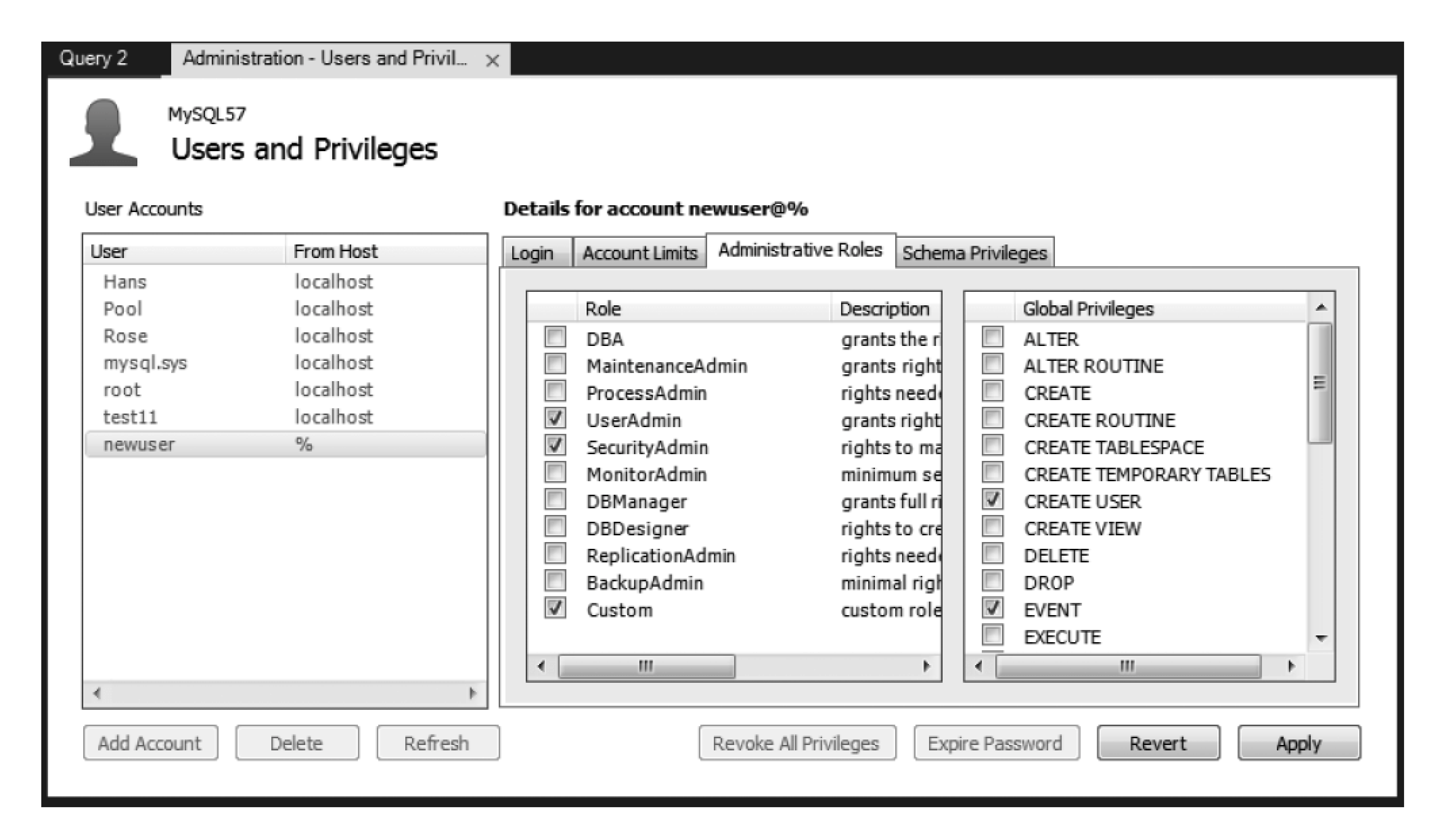


图 10-6　设置 test17 用户角色

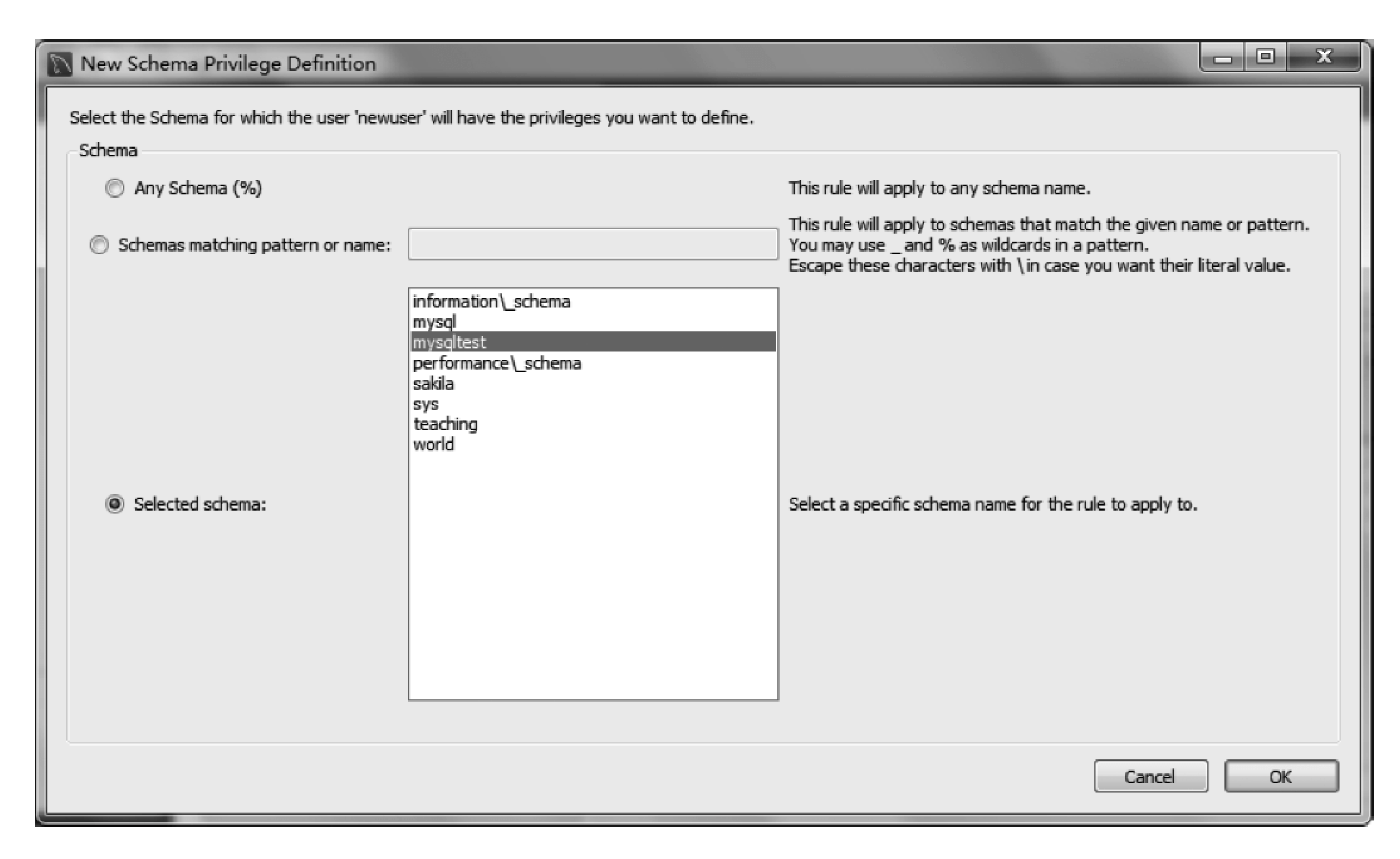


图 10-7　选择用户 test17 的数据库

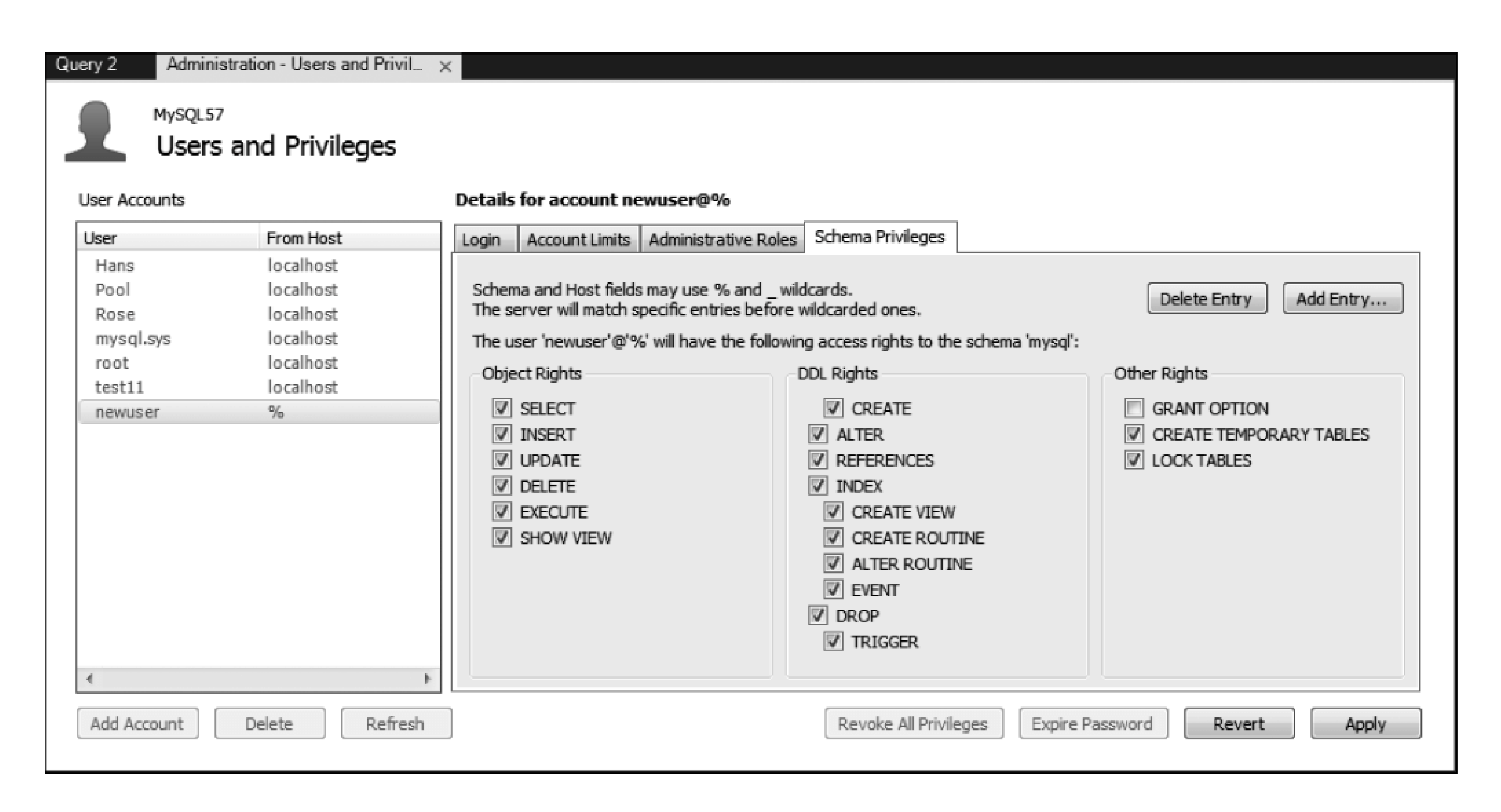


图 10-8　设置用户权限

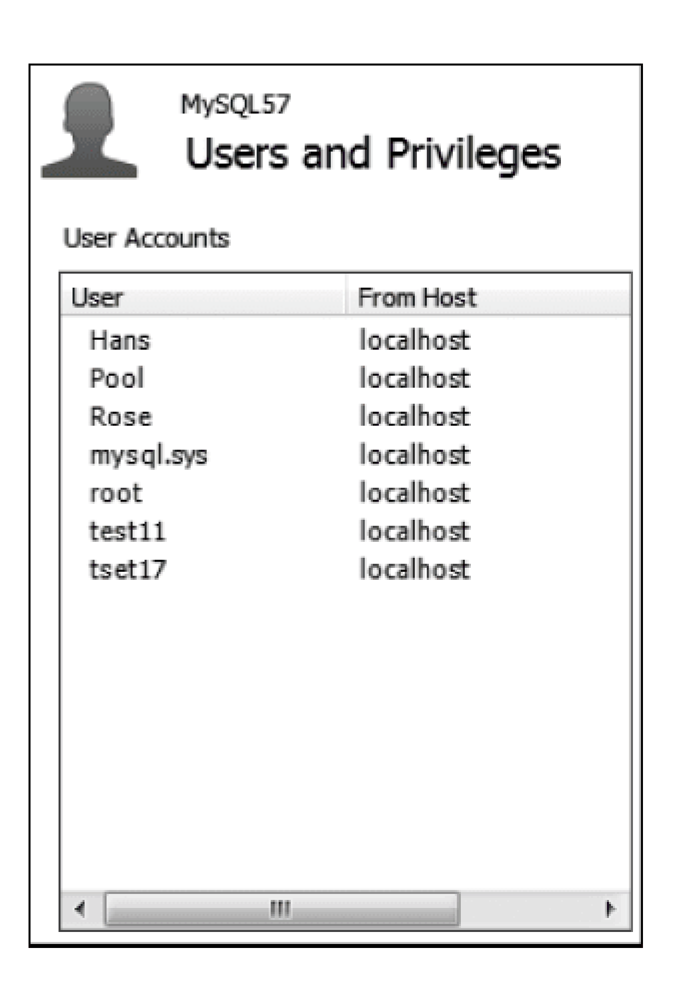


图 10-9　创建用户完成

10.3　权 限 管 理

权限管理主要是对登录到 MySQL 服务器的用户进行权限验证。所有用户的权限都存储在 MySQL 的权限表中。合理的权限管理能够保证数据库系统的安全，不合理的权限设置会给 MySQL 服务器带来安全隐患。

10.3.1　MySQL 的权限类型

MySQL 数据库中有多种类型的权限，这些权限都存储在 MySQL 数据库的权限表中。在 MySQL 启动时，服务器将这些数据库中的权限信息读入内存。

grant 和 revoke 命令用来管理访问权限，也可以用来创建和删除用户，但在 MySQL 5.7 中可以利用 create user 和 drop user 命令更容易地实现这些任务。

如果授权表拥有含有 mixed-case 数据库或表名称的权限记录，并且 lower_case_table_names 系统变量已设置，则不能使用 revoke 撤销权限，必须直接操纵授权表（当 lower_case_table_names 已设置时，grant 将不会创建此类记录，但是此类记录可能已经在设置变量之前被创建了。）

授予的权限可以分为多个层级：

（1）全局层级。全局权限适用于一个给定服务器中的所有数据库。这些权限存储在 mysql.user 表中。grant all on *.* 和 revoke all on *.* 只授予和撤销全局权限。

（2）数据库层级。数据库权限适用于一个给定数据库中的所有目标。这些权限存储在 mysql.db 和 mysql.host 表中。grant all on db_name.* 和 revoke all on db_name.* 只授予和撤销数据库权限。

（3）表层级。表权限适用于一个给定表中的所有列。这些权限存储在 mysql.tables_priv 表中。grant all on db_name.tbl_name 和 revoke all on db_name.tbl_name 只授予和撤销表权限。

（4）列层级。列权限适用于一个给定表中的单一列。这些权限存储在 mysql.columns_priv 表中。当使用 revoke 时，用户必须指定与被授权列相同的列。采用 select（col1，

col2,…)、insert(col1,col2,…)和 update(col1,col2,…)的格式实现。

(5) 子程序层级。create routine、alterroutine、execute 和 grant 等权限适用于已存储的子程序。这些权限可以被授予为全局层级和数据库层级。而且,除了 create routine 外,这些权限可以被授予为子程序层级,并存储在 mysql.procs_priv 表中。

grant 和 revoke 命令对于谁可以操作服务器及其内容的各个方面提供了多程度的控制,从谁可以关闭服务器,到谁可以修改特定表字段中的信息都能控制。表 10-1 列出了使用这些命令可以授予或撤回的常用权限。

表 10-1　grant 和 revoke 的常用管理权限

权　限	含　义
all [privileges]	设置除 grant option 之外的所有简单权限
alter	允许使用 alter table
alter routine	更改或取消已存储的子程序
create	允许使用 create table
create routine	创建已存储的子程序
create temporary tables	允许使用 create temporary table
create user	允许使用 create user, drop user, rename user 和 revoke all privileges
create view	允许使用 create view
delete	允许使用 delete
drop	允许使用 drop table
execute	允许用户运行已存储的子程序
file	允许使用 select…into outfile 和 load data infile
index	允许使用 create index 和 drop index
insert	允许使用 insert
lock tables	允许对用户拥有 select 权限的表使用 lock tables
process	允许使用 show full processlist
references	未被实施
reload	允许使用 flush
replication client	允许用户询问从属服务器或主服务器的地址
replication slave	用于复制型从属服务器(从主服务器中读取二进制日志事件)
select	允许使用 select
show databases	show databases 显示所有数据库
show view	允许使用 show create view
shutdown	允许使用 mysqladmin shutdown
super	允许使用 change master,kill,purge masterlogs 和 set global 语句,mysqladmin debug 命令;允许用户连接(一次),即使已达到 max_connections
update	允许使用 update
usage	"无权限"的同义词

10.3.2　授权管理


授权管理

授权就是为某个用户授予权限。在 MySQL 中,可以使用 grant 语句为用户授予权限。新创建的用户还没任何权限,不能访问数据库,不能做任何事情。针对不同用户对数据库的实际操作要求,分别授予用户对特定

表的特定字段、特定表、数据库的特定权限。

1. 利用 grant 语句给用户授权

在 MySQL 中使用 grant 关键字来为用户设置权限。必须是拥有 grant 权限的用户才可以执行 grant 语句。

grant 语句的基本语法格式如下：

```
grant priv_type[(column_list)][,priv_type[(column_list)]][,...n]
on {db_name.*|*.*|database_name.*|database_name.table_name}
to user[identified by [password] 'password']
[,user[identified by [password] 'password']] [,...n]
[with grant option];
```

说明：

(1) priv_type[(column_list)]：要设置的权限项。若授予用户所有的权限(all)，该用户为超级用户账户，具有完全的权限，可以做任何事情。

(2) {table_name|*|*.*|database_name.*|database_name.table_name}：对象类型项。可以是特定表、所有表、特定库或所有数据库。db_name.*表示特定数据库的所有表，*.*表示所有数据库。

(3) user[identified by [password] 'password']：是用户名和密码。

(4) with grant option：在授权时若带有 with grant option 语句，可以将该用户的权限转移给其他用户。

【例 10-4】 使用 grant 语句创建一个新用户 grantuser，密码为 grantpass。用户 grantuser 对所有的数据有查询、插入权限，并授予 grant 权限。

代码和运行结果如下：

```
mysql> grant select,insert on *.* to 'grantuser'@'localhost'
    -> identified by 'grantpass'
    -> with grant option;
      Query OK, 0 rows affected, 1 warning (0.47 sec)
```

【例 10-5】 使用 grant 语句将 teaching 数据库中 student 表的 delete 权限授予用户 grantuser。

代码和运行结果如下：

```
mysql> grant delete on teaching.student
    -> to 'grantuser'@'localhost';
      Query OK, 0 rows affected (0.15 sec)
```

【例 10-6】 授予 grantuser 在 student 表上的 studentno 列和 sname 列的 update 权限。

代码和运行结果如下：

```
mysql> grant update(studentno, sname)
    -> on student
    -> to grantuser@localhost;
      Query OK, 0 rows affected (0.04 sec)
```

【例 10-7】 授予用户 grantuser 为 teaching 数据库创建存储过程和存储函数权限。

代码和运行结果如下：

```
mysql > grant create routine on teaching. *
    -> to grantuser@localhost;
       Query OK, 0 rows affected (0.04 sec)
```

说明：

(1) 使用 grant 语句创建了用户的同时也完成授权。如果权限授予了一个不存在的用户，MySQL 会自动执行一条 create user 语句来创建这个用户，但必须为该用户指定密码。

(2) 对于列权限，权限的值只能取 select、insert 和 update。权限的后面需要加上列名。可以同时授予多个列权限，列名与列名之间用逗号分隔。

(3) 可以同时授予多个用户多个权限，权限与权限之间用逗号分隔，用户名与用户名之间用逗号分隔。

(4) 上述用 grant 语句进行授权，将会在授权表 db 中增加相应记录。

2. 利用 grant 语句实现权限转移

grant 语句的最后可以指定为 with grant option，则表示子句中指定的所有用户都有把自己所拥有的权限授予其他用户的权利，而不管其他用户是否拥有该权限。

【例 10-8】 授予 grantuser 用户 select、insert、update、delete、create、drop 权限，同时允许将其本身权限转移给其他用户。

代码和运行结果如下：

```
mysql > grant select, insert, update, delete, create, drop
    -> on teaching. *  to grantuser@localhost
    -> with grant option;
       Query OK, 0 rows affected (0.05 sec)
```

10.3.3 收回权限

收回权限就是取消已经赋予用户的某些权限。收回用户不必要的权限在一定程度上可以保证数据的安全性。权限收回后，用户账户的记录将从 db、host、tables_priv 和 columns_priv 表中删除，但是用户账户记录仍然在 user 表中保存。收回权限利用 revoke 语句来实现，语法格式有两种，一种是收回用户指定的权限，另一种是收回用户的所有权限。

1. 收回指定权限

收回用户指定权限的基本语法如下：

```
revoke priv_type[(column_list)][,priv_type[(column_list)]][,…n]
      on{table_name| * | * . * |database_name. * |database_name.table_name}
      from 'username'@'hostname'[,'username'@'hostname'][,…n];
```



权限管理

【例 10-9】 收回 grantuser 用户对 teaching 数据库中 student 表的 update 权限。

代码和运行结果如下：

```
mysql > revoke update on teaching. student
    -> from grantuser@localhost;
       Query OK, 0 rows affected (0.05 sec)
```

2. 收回所有权限

收回用户所有权限的基本语法如下：

```
revoke all privileges,grant option
from 'username'@'hostname'[,'username'@'hostname'][,…n];
```

【例 10-10】 使用 revoke 语句收回 grantuser 用户的所有权限，包括 grant 权限。

代码和运行结果如下：

```
mysql> revoke all privileges,grant option
    -> from grantuser@localhost;
       Query OK, 0 rows affected (0.00 sec)
```

10.3.4 查看权限

show grants 语句可以显示指定用户的权限信息。

使用 show grants 查看账户权限信息的基本语法格式如下：

```
show grants for 'username'@'hostname';
```

例如，使用 show grants 语句查看 grantuser 用户的权限信息：

```
mysql> show grants for grantuser@localhost;
```

10.3.5 限制权限

with 子句也可以通过下列参数实现对一个用户授予使用限制，其中，count 表示次数。

(1) max_queries_per_hour count 表示每小时可以查询数据库的次数。

(2) max_connections_per_hour count 表示每小时可以连接数据库的次数。

(3) max_updates_per_hour count 表示每小时可以修改数据库的次数。

【例 10-11】 授予 grantuser 每小时只能处理一条 select 语句的权限。

代码和运行结果如下：

```
mysql> grant select
    -> on teaching.student
    -> to grantuser@localhost
    -> with max_queries_per_hour 1;
       Query OK, 0 rows affected, 1 warning (0.00 sec)
```

【例 10-12】 授予 grantuser 每小时可以发出的查询数为 20 次，每小时可以连接数据库 5 次，每小时可以发出的更新数为 10 次。

代码和运行结果如下：

```
mysql> grant all on *.* to grantuser@localhost
    -> identified by 'grantpass'
    -> with max_queries_per_hour 20
    -> max_updateS_per_hour 10
    -> max_connections_per_hour 5;
       Query OK, 0 rows affected, 1 warning (0.00 sec)
```

10.4 MySQL 数据库安全常见问题

10.4.1 权限更改何时生效

MySQL 服务器启动时以及使用 grant 和 revoke 语句时，服务器会自动读取 grant 表。这就为手动更新这些权限表提供了方法。当手动更新权限表时，MySQL 服务器将不会自动监测到这些修改的权限。

有 3 种方法可以实现让服务器完善这些修改权限，使之生效。最常使用的更新权限的方法是在 MySQL 命令提示符下(必须以管理员的身份登录进入)输入如下命令：

```
flush privileges;
```

或者，还可以在操作系统中运行：

```
mysqladmin flush-privileges
```

或者是：

```
mysqladmin reload
```

此后，当用户下次再连接的时候，系统将检查全局级别权限；当下一个命令被执行时，将检查数据库级别的权限；而表级别和列级别权限将在用户下次请求的时候被检查。

10.4.2 设置账户密码

(1) 在 DOS 命令窗口中指定密码。利用 mysqladmin 命令重设服务器为 host_name，且用户名为 user_name 的用户的密码，新密码为 newpassword。

```
mysqladmin -u user_name -h host_name password 'newpassword'
```

(2) 通过 set password 命令设置用户的密码。只有以 root 用户或有更新 MySQL 数据库权限的用户的身份登录，才可以更改其他用户的密码。

```
mysql> set password for 'pool'@'%' = password('base123456');
```

如果没有以匿名用户连接，省略 for 子句便可以更改自己的密码：

```
mysql> set password = password('base123456');
```

(3) 使用 grant usage 语句指定账户密码。在全局层级下使用 grant usage 语句(在 *.*)指定某个账户的密码，而不影响账户当前的权限：

```
mysql> grant usage on *.* to 'pool'@'%' identified by 'base123456';
```

(4) 在创建新账户时建立密码，要为 password 列提供一个具体值：

```
mysql -u root mysql insert into user (Host,User,Password)
   values('%','pool33',password('base123456'));
mysql> flush privileges;
```

(5) 更改已有账户的密码,要应用 update 语句来设置 password 列值:

```
mysql  - u root mysql
update user set Password = password('base123456')
where Host = '%' and User = 'test11';
mysql > flush privileges;
```

说明:

(1) 如果使用 grant … identified by 语句或 mysqladmin password 命令设置密码,它们均会自动加密密码。在这种情况下,不需要使用 password()函数对密码进行加密。

(2) 当使用 set password、insert 或者 update 指定账户的密码时,必须用 password()函数对它进行加密(唯一的特例是如果密码为空,则不需要使用 password())。之所以使用 password()是因为 user 表以加密方式保存密码,而不是明文。如果采用没有进行加密的方式设置密码,结果是密码 base123456 保存到 user 表后没有加密。当用户使用该密码连接服务器时,连接使用的密码值将被加密,并同保存在 user 表中的密码进行比较。因此比较将失败,服务器拒绝连接。

10.4.3 使密码更安全

MySQL 授权 user 表定义初始 MySQL 用户账户和访问权限。在 Windows 中,MySQL 一般创建了两个 root 账户。一个 root 账户用来从本机连接 MySQL 服务器,具有所有权限;另一个允许从任何主机连接,具有 test 数据库或其他以 test 开始的数据库的所有权限。

初始账户均没有密码,因此任何人可以用 root 账户不用任何密码来连接 MySQL 服务器。一般情况下,MySQL 创建了两个匿名用户账户,每个账户的用户名均为空。匿名账户没有密码,因此任何人可以使用匿名账户来连接 MySQL 服务器。在 Windows 中,一个匿名账户用来从本机进行连接。也具有所有权限,同 root 账户一样。另一个可以从任何主机上连接,具有 test 数据库或其他以 test 开始的数据库的所有权限。为了更好地保证 MySQL 数据库的安全,可以采取如下措施。

(1) 为 root 账户指定密码,使用 set password 语句为匿名账户指定密码。使用 set password 语句指定 root 账户密码的命令如下:

```
mysql > set password for root@localhost = password('123456');
mysql > set password for ''@'localhost' = password('1234546');
```

或者使用 update 语句修改 root 账户密码,要注意使用 password()函数为密码加密的限制。使用 update 更新密码后,必须让服务器用 flush privileges 重新读授权表。命令如下:

```
mysql > update mysql.user set Password = password('123456')
    -> where User = 'root';
mysql > update mysql.user set Password = password('123456')
    -> where User = '';
mysql > flush privileges;
```

(2) 直接删除匿名账户,可以避免匿名访问用户的危害。命令如下:

```
mysql> delete from mysql.user where User = '';
```

10.4.4 要确保 MySQL 的安全的注意事项

(1) 管理员在管理用户级别时,切忌不能将 mysql.user 表的访问权限授予任何一般账户。

(2) 如果从非交互式方式下运行一个脚本调用一个客户端,就没有从终端输入密码的机会。其最安全的方法是让客户端程序提示输入密码或在适当保护的选项文件中指定密码。

(3) 可以采用下面的命令模式来连接服务器,以此来隐藏我们的密码。命令如下:

```
mysql -u laisone -p db_name
Enter password: ********
```

其中,"*"字符指示输入密码的地方,输入的密码对其他用户是不可见的。

(4) 审计服务器的用户账户。当已有的服务器作为公司的数据库主机时,要确保禁用所有非特权用户,或者最好是全部删除。虽然 MySQL 用户和操作系统用户完全无关,但他们都要访问服务器环境,仅凭这一点就可能会有意地破坏数据库服务器及其内容。为完全确保在审计中不会有遗漏,可以考虑重新格式化所有相关的驱动器,并重新安装操作系统。

(5) 设置 MySQL 的 root 用户密码。对所有 MySQL 用户使用密码。客户端程序不需要验证运行它的人员的身份。对于客户端/服务器应用程序,用户可以指定客户端程序的用户名。例如,如果 abc_user 没有密码,任何人可以简单地用 mysql -u other_user db_name 冒充他人调用 mysql 程序进行连接。如果所有用户账户均存在密码,使用其他用户的账户进行连接将困难得多。

(6) 及时下载安装补丁软件。为操作系统和安装软件下载安装补丁软件是屏蔽恶意用户攻击的常用方法,否则,即使恶意用户没有多少攻击经验,也可以毫无阻碍地攻击未打补丁的服务器。

(7) 禁用所有不使用的系统服务。始终要注意在将服务器放入网络之前,已经消除所有不必要的潜在服务器攻击途径。

(8) 关闭未使用的端口。虽然关闭未使用的系统服务是减少成功攻击可能性的好方法,不过还可以通过关闭未使用的端口来添加第二层安全。对于专用的数据库服务器,如果不希望在指定端口有数据通信,就关闭这个端口。除了在专用防火墙工具或路由器上做这些调整之外,还可以考虑利用操作系统的防火墙。

10.5 小　　结

本章介绍了 MySQL 数据库的权限表、账户管理、权限管理的内容。其中,密码管理、授权、收回权限等内容涉及 MySQL 数据库的安全,在实际应用中非常重要。至于授权时需要确定给用户分配什么权限,这需要根据实际情况来决定。学习本章后,需要具体掌握如下内容:

- MySQL 数据库中的表与其他任何关系表没有区别，都可以通过典型的 SQL 命令修改其结构和数据。
- MySQL 账户的创建、删除、重命名等操作过程。
- 管理访问权限的 grant 和 revoke 命令格式、应用方法和主要用途。
- 确保 MySQL 数据库的安全的注意事项。

习　题　10

1. 选择题

(1) 在 MySQL 中，可以使用________语句来为指定数据库添加用户。

A. revoke　　B. grant　　C. insert　　D. create

(2) MySQL 中存储用户全局权限的表是________。

A. tables_priv　　B. procs_priv　　C. columns_priv　　D. user

(3) 下列命令中，________命令用于撤销 MySQL 用户对象权限。

A. revoke　　B. grant　　C. deny　　D. create

(4) 给名字是 liping 的用户分配对数据库 teaching 中的 student 表的查询和插入数据权限的语句是________。

A. grant select,insert on teaching. student for 'liping'@'localhost'

B. grant select,insert on teaching. student to 'liping'@'localhost'

C. grant 'liping'@'localhost' to select,insert for teaching. student

D. grant 'liping'@'localhost' to teaching. student on select,insert

(5) 修改自己的 MySQL 服务器密码的命令是________。

A. mysql　　B. grant

C. set password　　D. change password

2. 思考题

(1) 简述 MySQL 在对象上进行权限设置时，用户和权限的关系。

(2) 在 MySQL 中可以授予的权限有哪几个层次？

(3) 在 MySQL 的权限授予语句中，可用于指定权限级别的值有哪几类格式？

(4) 在数据加密过程中，password()函数有什么作用？

3. 上机练习题(本题利用 teaching 数据库进行操作)

(1) 使用 grant 语句创建一个新用户 ex_user，密码为 pass123。用户 ex_user 对当前数据库中的所有表有查询、插入权限，并授予 grant 权限。

(2) 利用 grant 语句将 teaching 数据库中 student 表的 delete 权限授予用户 ex_user。

(3) 收回 ex_user 用户对 teaching 数据库中 student 表的 delete 权限。

(4) 使用 revoke 语句收回 ex_user 用户的所有权限，包括 grant 权限。

(5) 假定当前系统中不存在用户 swming，请编写一段 SQL 语句，要求创建这个新用户，并为其设置对应的系统登录口令“my123”，同时授予该用户在数据库 teaching 的表 course 上拥有 select 和 update 的权限。

第 11 章　备份与恢复

数据库的安全性和完整性是确保实现数据的可靠性、精确性和高效性的重要技术手段。为了保证数据的安全,防止意外事件的发生,需要制度化地定期对数据进行备份。如果数据库系统数据遭到破坏,就可以使用备份好的数据进行数据还原,将损失降到最小。另外数据表之间的数据导入与导出技术,也为数据管理提供了可靠的备份功能。

本章主要介绍数据损失的原因,以及数据备份和数据恢复的方法,并要求理解数据库迁移、数据的导入与导出的方法。

11.1　备份和恢复概述

数据备份和恢复是数据库管理中最常用的操作。备份和恢复的目的就是将数据库中的数据进行导出,生成副本,然后在系统发生故障后能够恢复全部或部分数据。数据备份就是制作数据库结构、对象和数据的备份,以便在数据库遭到破坏时,或因需求改变而能够把数据库还原到改变以前时的状态。数据恢复就是指将数据库备份加载到系统中。数据备份和恢复可以用于保护数据库的关键数据,在系统发生错误或者因需求改变时,利用备份的数据可以恢复数据库中的数据。

1. 数据丢失的原因

对于生产数据库来说,数据的安全性是至关重要的,任何数据的丢失和危险都可能给生产带来严重的损失。例如,金融行业数据库系统存储着客户账户的重要信息,绝对不允许出现故障和数据破坏。为了保证数据的安全,需要定期对数据进行备份。

制定各种故障和灾难的恢复计划,应该预计到各种形式的潜在灾难,并针对具体情况制定恢复计划。例如,数据库系统在运行过程中可能出现运行故障,计算机系统由于出现操作失误或系统故障、自然灾害、计算机病毒或者物理介质故障等。

在数据库系统生命周期中可能发生的灾难主要分为三类。

(1) 系统故障。系统故障一般是指硬件故障或软件错误。

(2) 事务故障。事务故障是指事务运行过程中,没有正常提交就产生的故障。MySQL还可以通过重启服务来处理该故障。

(3) 介质故障。由于物理介质发生读写错误,或者管理员在操作过程中不慎删除一些重要数据或日志文件,就会产生介质故障。一般来说,介质故障需要数据库管理员手工进行恢复,恢复时需要在发生故障前的数据库备份和日志备份。

2. 数据备份的分类

(1) 按备份时服务器是否在线划分。

- 热备份。热备份是指数据库在线时服务正常运行的情况下进行数据备份。

- 温备份。温备份是指进行数据备份时数据库服务正常运行，但数据只能读不能写。
- 冷备份。冷备份是指数据库已经正常关闭的情况下进行的数据备份，当正常关闭时会提供一个完整的数据库。

(2) 按备份的内容划分。

- 逻辑备份。逻辑备份是指使用软件技术从数据库中导出数据并写入一个输出文件，该文件格式一般与原数据库的文件格式不同，只是原数据库中数据内容的一个映像。逻辑备份支持跨平台，备份的是 SQL 语句(DDL 和 insert 语句)，以文本形式存储。在恢复的时候执行备份的 SQL 语句实现数据库数据的重现。
- 物理备份。物理备份是指直接复制数据库文件进行的备份，与逻辑备份相比，其速度较快，但占用空间比较大。

(3) 按备份涉及的数据范围来划分。

- 完整备份。完整备份是指备份整个数据库。这是任何备份策略中都要求完成的第一种备份类型，因为其他所有备份类型都依赖于完整备份。换句话说，如果没有执行完整备份，就无法执行差异备份和增量备份。
- 增量备份。数据库从上一次完全备份或者最近一次的增量备份以来改变的内容的备份。
- 差异备份。差异备份是指将从最近一次完整数据库备份以后发生改变的数据进行备份。差异备份仅捕获自该次完整备份后发生更改的数据。

备份是一种十分耗费时间和资源的操作，不能频繁操作。应该根据数据库使用情况确定一个适当的备份周期。

3. 备份的时机

备份数据库的时机和频率取决于可接受的数据丢失量和数据库活动的频繁程度。需要决定从每种灾难中进行数据还原的合理时间长度，根据灾难类型和数据库的大小不同，所需的最短数据还原时间也会不同。

用户应当定期地备份用户数据库。可以从下列几方面考虑备份的时机。

(1) 创建数据库或为数据库填充了数据以后，用户应该备份数据库。

(2) 创建索引后备份数据库。

(3) 清理事务日志后备份数据库。当执行了清理事务日志的语句后，应该备份数据库。在清理之后，事务日志将不包含数据库的活动记录，也不能用来还原数据库。

(4) 执行了无日志操作后也应该备份数据库。

4. 数据恢复需要注意的问题

数据恢复就是在数据库的一定生命周期的某一时刻还原数据。管理员的非法操作和计算机的故障都会破坏数据库文件。当数据库遭到这些意外时，可以通过备份文件将数据库还原到备份时的状态。这样可以将损失降低到最小。当计划从各种潜在的灾难中恢复时，需要考虑相关的问题，并为各种可能性做准备。例如，一个包含数据文件的磁盘出现故障，就应该考虑下列问题。

(1) 关闭数据库会造成什么后果？

(2) 替换损坏的数据磁盘并用数据库备份还原数据的时间可否接受？

(3) 如何使数据库不会由于单个磁盘的故障而无法使用？

(4) 用数据库备份还原数据的实际时间是多少？

(5) 更频繁地备份数据库是否会显著地减少还原时间?

5. 数据恢复的方法

数据恢复就是当数据库出现故障时,将备份的数据库加载到系统,从而使数据库恢复到备份时的正确状态。MySQL 有 3 种保证数据安全的方法。

(1) 数据库备份:通过导出数据或者表文件的拷贝来保护数据。

(2) 二进制日志文件:保存更新数据的所有语句。

(3) 数据库复制:MySQL 内部复制功能。建立在两个或两个以上服务器之间,通过设定它们之间的主从关系来实现。其中一个作为主服务器,其他的作为从服务器。在此主要介绍前两种方法。

恢复是与备份相对应的系统维护和管理操作。系统进行恢复操作时,先执行一些系统安全性的检查,包括检查所要恢复的数据库是否存在、数据库是否变化及数据库文件是否兼容等,然后根据所采用的数据库备份类型采取相应的恢复措施。

数据备份是数据库管理员的工作。系统意外崩溃或者硬件的损坏都可能导致数据库的丢失,因此 MySQL 管理员应该定期对数据库进行备份,使得在意外情况发生时,尽可能减少损失。

11.2 数据备份

11.2.1 使用 mysqldump 命令备份

MySQL 提供了很多免费的客户端程序和实用工具,在 MySQL 目录下的 bin 子目录中存储着这些客户端程序。不同的 MySQL 客户端程序可以连接服务器以访问数据库或执行不同的管理任务。



利用 mysqldump 命令备份

mysqldump 命令就是 MySQL 提供的一个非常有用的数据库备份工具。该实用程序存储在 C:\Program Files\MySQL\MySQLServer 5.7\bin 文件夹中。mysqldump 命令执行时,可以将数据库备份成一个文本文件,该文件中实际上是包含了多个 create 和 insert 语句,使用这些语句可以重新创建表和插入数据。表的结构和表中的数据将存储在生成的文本文件中。

mysqldump 命令的工作原理很简单,即先查出需要备份的表的结构,再在文本文件中生成一个 create 语句。然后,将表中的所有记录转换成一条 insert 语句。这些 create 语句和 insert 语句都是还原时使用的。还原数据时就可以使用其中的 create 语句来创建表。使用其中的 insert 语句来还原数据。

默认 mysqldump 导出的.sql 文件中不但包含了表数据,还包括导出数据库中所有数据表的结构信息。另外,使用 mysqldump 导出的 SQL 文件如果不带绝对路径,默认是保存在 bin 目录下的。

1. 备份数据库或表

mysqldump 备份数据库或表的基本语法格式如下:

```
mysqldump -u user -h host -ppassword
 --databasename[all-databases][tablename=,[tablename=...]]> filename.sql;
```

说明：

（1）-h 后面是主机名，如是本地主机登录，此项可忽略。

（2）使用 mysqldump 要指定用户名和密码。其中，-u 后面是用户名，-p 后面是密码，-p 选项与密码之间不能有空格。

（3）--databasename[tablename,[tablename...]]是选项，选项很多，下面只列出几个常用的选项。

- databasename：表示备份数据库。
- --all-databases：表示备份所有数据库。
- --tablename=：表示数据和创建表的 SQL 语句分开备份成不同的文件。

（4）filename. sql 是输出文件，可以指定路径。

（5）mysqldump 命令中各参数的含义可以通过运行帮助命令 mysqldump-help，获得特定版本的完整参数列表。

【例 11-1】 使用 mysqldump 命令备份数据库 mysqltest 中的所有表。

代码和运行结果如下：

```
mysqldump -u root -p mysqltest > d:/bak/mysqltestbak.sql
Enter password: ******
```

输入密码后，MySQL 便对数据库进行了备份，在 d:\bak 文件夹下查看备份的文件，使用文本查看器打开文件可以看到其文件内容，查看文件 mysqltestbak. sql 的内容，如图 11-1 所示。



图 11-1　查看 mysqltestbak. sql 文本文件

【例 11-2】 使用 mysqldump 命令备份数据库中的 student 表和 score 表。

代码和运行结果如下：

```
mysqldump -u root -p teaching student score > d:/bak/teaching_ss.sql
Enter password: ******
```

【例 11-3】 使用 mysqldump 命令备份数据库中的 course 表。

代码和运行结果如下：

```
mysqldump -u root -p teaching course > d:/bak/course.sql
Enter password: ******
```

说明：

利用 mysqldump 命令备份的表文件中，主要包括创建该表的 create 命令代码和插入该表数据的 insert 命令。

2. 备份多个数据库

使用 mysqldump 备份多个数据库，需要使用--databases 参数。

基本语法格式如下：

```
mysqldump -u user -h host -p --databases databasename[databasename...]]> filename.sql;
```

使用--databases 参数之后，必须指定至少一个数据库的名称，多个数据库之间用空格隔开。

【例 11-4】 使用 mysqldump 命令备份数据库 teaching 和 mysqltest。

代码和运行结果如下：

```
mysqldump -u root -p --databases teaching mysqltest > d:/bak/teach_test.sql
Enter password: ******
```

3. 查看备份文件

mysqldump 能够生成移植到其他机器的文本文件，甚至可移植到那些有不同硬件结构的机器上，mysqldump 产生的输出可在以后用作 MySQL 的输入来重建数据库。例如：

```
mysqldump -uroot -p123456 --databases teaching > d:/bak/teach.txt
```

在文本文件 teach.txt 中输出了表创建、表数据插入，以及存储过程、存储函数、触发器、事件等对象的创建语句。这些语句可作为输入来创建 MySQL 数据库，如图 11-2 所示。


teach.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
--
-- Table structure for table `score`
--

DROP TABLE IF EXISTS `score`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `score` (
  `studentno` char(11) NOT NULL,
  `courseno` char(6) NOT NULL,
  `daily` float(3,1) DEFAULT '0.0',
  `final` float(3,1) DEFAULT '0.0',
  PRIMARY KEY (`studentno`,`courseno`),
  KEY `sc_index` (`studentno`,`courseno`)
) ENGINE=InnoDB DEFAULT CHARSET=gb2312;
/*!40101 SET character_set_client = @saved_cs_client */;
```

第 268 行，第 1 列

图 11-2　查看文本文件

11.2.2　直接复制整个数据库目录

因为 MySQL 表保存为文件方式，所以可以直接复制 MySQL 数据库的存储目录及文

件进行备份。这种方法最简单，速度也最快。使用该方法时，最好先将服务器停止，这样可以保证在复制期间数据不会发生变化。

这种方法虽然简单快速，但不是最好的备份方法。因为实际情况可能不允许停止MySQL服务器。而且，这种方法对InnoDB存储引擎的表不适用。对于MyISAM存储引擎的表，这样备份和还原很方便。但是还原时最好是相同版本的MySQL数据库，否则可能会存在文件类型不同的情况。

11.2.3 使用mysqlhotcopy工具快速备份

如果备份时不能停止MySQL服务器，可以采用mysqlhotcopy工具。mysqlhotcopy工具的备份方式比mysqldump命令快。

mysqlhotcopy工具是一个Perl脚本，主要在Linux操作系统下使用。mysqlhotcopy工具使用lock tables、flush tables和cp来进行快速备份。

mysqlhotcopy工具的工作原理是：先将需要备份的数据库加上一个读操作锁，然后用flush tables将内存中的数据写回到硬盘上的数据库中，最后把需要备份的数据库文件复制到目标目录。

使用mysqlhotcopy的命令如下：

```
[root@localhost ~]# mysqlhotcopy [option] dbname1 dbname2 ... backupDir/
```

相关参数的含义可以通过网络查询了解。

11.3 数据恢复

恢复数据库，就是让数据库根据备份的数据回到备份时的状态。当数据丢失或意外破坏时，可以通过数据恢复已经备份的数据，尽量减少数据丢失和破坏造成的损失。

11.3.1 使用MySQL命令恢复数据


利用MySQL命令恢复数据

管理员通常使用mysqldump命令将数据库中的数据备份成一个文本文件。通常这个文件的后缀名是.sql。需要还原时，可以使用MySQL命令来还原备份的数据。

对于使用mysqldump命令备份后形成的.sql文件，可以使用MySQL命令导入到数据库中。备份的.sql文件中包含create、insert语句，也可能包含drop语句。MySQL命令可以直接执行文件中的这些语句。

MySQL命令恢复数据的语法格式如下：

```
mysql -u user -p [databasename]< filename.sql;
```

【例 11-5】 使用MySQL命令将备份文件mysqltestbak.sql恢复到数据库中。

代码和运行结果如下：

```
mysql -u root -p mysqltest < d:\bak\mysqltestbak.sql
Enter password: ******
```

执行语句前，必须先在MySQL服务器中创建mysqltest数据库，如果不存在此数据库，

在数据恢复过程中会出错。命令执行成功之后,mysqltestbak. sql 文件中的语句就会在指定的数据库中恢复以前的数据。

11.3.2 使用 source 恢复表和数据库



利用 source 命令恢复数据

MySQL 最常用的数据库导入命令就是 source,source 命令的用法非常简单,首先进入 MySQL 数据库的命令行管理界面,然后选择需要导入的数据库。

使用 source 命令能够将备份好的. sql 文件导入到 MySQL 数据库中。

1. 恢复表

【例 11-6】 删除 course 表的数据,用 source 命令恢复。

代码和运行结果如下:

```
#尝试删除 course 表的数据
mysql > use teaching;
        Database changed
mysql > delete from course;
        Query OK, 13 rows affected (0.11 sec)
```

利用放在"d:/ bak"路径下 course 的备份文件 course. sql,使用 source 命令把备份好的文件导入进行恢复:

```
mysql > source d:/bak/course.sql;
        Query OK, 0 rows affected (0.00 sec)
        …
        Query OK,13 rows affected (0.03 sec)
        Records:13 Duplicates: 0 Warnings: 0
        …
        Query OK, 0 rows affected (0.00 sec)
```

2. 恢复数据库

如果已登录 MySQL 服务器,还可以使用 source 命令导入. sql 文件。

source 语句的语法如下:

```
source filename.sql
```

【例 11-7】 使用 source 命令将备份文件 mysqltestbak. sql 恢复到数据库中。

代码和运行结果如下:

```
mysql > use mysqltest;
        database changed
mysql > source d:\bak\mysqltestbak.sql
        Query OK, 0 rows affected (0.00 sec)
        …
        Query OK, 0 rows affected (0.00 sec)
```

说明:

(1) 用 source 语句导入包含已备份好的. sql 文件,可以恢复整个数据库或某张表。

(2) 使用 source 命令必须进入 MySQL 控制台并进入到待恢复的数据库。

(3) 如果数据库已删除,由于没办法进入数据库,可以先建一个同名的空数据库,然后

用 use 命令使用该数据库，再用 source 命令进行恢复。

(4) 可以直接用 source 命令导入备份文件进行恢复。

(5) 在导入数据前，可以先确认编码，如果不设置可能会出现乱码。

```
mysql > set names gb2312;
mysql > source d:/backup/mysqltestbak.sql;
```

11.3.3 直接复制到数据库目录

如果数据库通过复制数据库文件备份，可以直接复制备份的文件到 MySQL 数据目录下实现还原。通过这种方式还原时，必须保证备份数据的数据库和待还原的数据库服务器的主版本号相同。而且这种方式只对 MyISAM 存储引擎的表有效，对于 InnoDB 存储引擎的表不可用。

执行还原前要关闭 MySQL 服务，将备份的文件或文件夹覆盖 MySQL 的 data 文件夹，然后再启动 MySQL 服务。对于 Linux/UNIX 操作系统来说，复制完文件需要将文件的用户和组更改为 MySQL 运行的用户和组，通常用户是 MySQL，组也是 MySQL。

11.4 数据库迁移

数据库迁移就是指将数据库从一个系统移动到另一个系统上。数据库迁移的原因是多种多样的。可能是因为升级了计算机，或者是部署开发的管理系统，或者升级了 MySQL 数据库。甚至是换用其他的数据库。根据上述情况，可以将数据迁移大致分为以下 3 类。

- 需要安装新的数据库服务器。
- MySQL 版本更新。
- 数据库管理系统的变更(如从 Microsoft SQL Server 迁移到 MySQL)。

数据库迁移可以使用一些工具，例如在 Windows 系统下，可以使用 MyODBC 实现 MySQL 和 SQL Server 之间的迁移。MySQL 官方提供的工具 MySQL Migration Toolkit 也可以在不同数据库之间进行数据迁移。

11.4.1 相同版本的 MySQL 数据库之间的迁移

相同版本的 MySQL 数据库之间的迁移就是在主版本号相同的 MySQL 数据库之间进行数据库移动。这种迁移的方式最容易实现。

相同版本的 MySQL 数据库之间进行数据库迁移的原因很多。通常的原因是换了新的机器，或者是装了新的操作系统。还有一种常见的原因就是将开发的管理系统部署到工作机器上。因为迁移前后 MySQL 数据库的主版本号相同，所以可以通过复制数据库目录来实现数据库迁移。但是，只有数据库表都是 MyISAM 类型的才能使用这种方式。

11.4.2 不同版本的数据库之间的迁移

不同版本的 MySQL 数据库之间进行数据迁移通常是 MySQL 升级的原因。例如，原来很多服务器使用 MySQL 5.0 数据库。MySQL 5.7 的版本推出以后，改进了 MySQL 5.0

版本的很多缺陷。因此需要将 MySQL 数据库升级到 MySQL 5.7 版本。这样就需要进行不同版本的 MySQL 数据库之间进行数据迁移。

高版本的 MySQL 数据库通常都会兼容低版本，因此可以从低版本的 MySQL 数据库迁移到高版本的 MySQL 数据库。对于 MyISAM 类型的表可以直接复制，也可以使用 mysqlhotcopy 工具。但是 InnoDB 类型的表不可以使用这两种方法。最常用的办法是使用 mysqldump 命令来进行备份，然后通过 MySQL 命令将备份文件还原到目标 MySQL 数据库中。但是，高版本的 MySQL 数据库很难迁移到低版本的 MySQL 数据库。因为高版本的 MySQL 数据库可能有一些新的特性，这些新特性是低版本 MySQL 数据库所不具有的。数据库迁移时要特别小心，最好使用 mysqldump 命令来进行备份，避免迁移时造成数据丢失。

11.4.3 不同类型的数据库之间的迁移

不同数据库之间迁移是指从其他类型的数据库迁移到 MySQL 数据库，或者从 MySQL 数据库迁移到其他类型的数据库。例如，某个网站原来使用 Oracle 数据库。因为运营成本太高等诸多原因，希望改用 MySQL 数据库。或者，某个管理系统原来使用 MySQL 数据库，因为某种特殊性能的要求，希望改用 Oracle 数据库。这样的不同数据库之间的迁移也经常会发生。但是这种迁移没有普通适用的解决办法。

MySQL 以外的数据库也有类似 mysqldump 这样的备份工具，可以将数据库中的文件备份成 SQL 文件或普通文本。但是，因为不同数据库厂商没有完全按照 SQL 标准来设计数据库。这就造成了不同数据库使用的 SQL 语句的差异。例如，Oracle 数据库软件使用的是 PL/SQL 语言，微软的 SQL Server 软件使用的是 Transact-SQL 语言。PL/SQL 语言和 Transact-SQL 语言中包含了非标准的 SQL 语句。这就造成了 Oracle、SQL Server 和 MySQL 的 SQL 语句不能兼容。

11.4.4 将数据库转移到新服务器

将数据库转移到新服务器的语法格式如下：

```
mysqldump –uusername –ppassword databasename |mysql –host=hostname –c databasename
```

mysqldump 还可以支持下列选项，例如：

(1) -add-locks：在每个表导出之前增加 lock tables 并且之后 unlock table(为了使得更快地插入到 MySQL)。

(2) -add-drop-table：在每个 create 语句之前增加一个 drop table。

(3) -allow-keywords：允许创建是关键词的列名字。

(4) -c,-complete-insert：使用完整的 insert 语句(用列名字)。

(5) -c,-compress：如果客户和服务器均支持压缩，压缩两者间所有的信息。

(6) -delayed：用 insert delayed 命令插入行。

(7) -e,-extended-insert：使用全新多行 insert 语法(给出更紧缩并且更快的插入语句)。

11.5 表的导入与导出

MySQL 数据库中的表可以导出成文本文件、xls、xml 或者 html 格式的文件。相应的文本文件也可以导入 MySQL 数据库中。

MySQL 数据库中的数据可以导出到外部存储文件中，在数据库的日常维护中，经常需要进行表的导出和导入的操作。MySQL 数据库中的数据可以导出为 sql 文本文件、xml 文件、txt 文件、xls 文件或 html 文件。同样，这些导出文件也可以导入到 MySQL 数据库中。

11.5.1 用 select...into outfile 导出文件

MySQL 中，可以使用 select...into outfile 语句将表的内容导出成各种格式的文件。

select...into outfile 语句基本语法形式如下：

```
select[columnlist]from table[where condition]into outfile'filename'[options];
```

说明：

(1) 该语句的作用是将表中 select 语句选中的行写入到一个文件中。文件默认在服务器主机上创建，并且原文件将被覆盖。

(2) into outfile 'filename'：将前面 select 语句的查询结果导出到文件名为 filename 的外部文件中。导出文件格式可以是文本文件、xls、xml 或者 html 格式的文件等。

(3) [options]为可选参数项，部分语法包含两个自选的子句：fields 子句和 lines 子句，其作用是决定数据行在文件中存放的格式。options 的格式为：

```
-- options 参数
  fields terminated by 'value'
  fields [optionally] enclosed by 'value'
  fields escaped by 'value'
  lines starting by 'value'
  lines terminated by 'value';
```

各参数可能的取值为：

- fields terminated by 'value'：设置字段之间的分隔符，可以为单个或多个字符，默认制表符为“\t”。
- fields [optionally] enclosed by 'value'：设置字段的包围字符，只能为单个字符，若使用 optionally 选项，则只能 char 和 varchar 等字符被包括。
- fields escaped by 'value'：设置如何写入或读入特殊字符，只能为单个字符，即设置转义字符，默认值为“\”。
- lines starting by 'value '：设置每行数据开头的字符，可以为单个或多个字符，默认情况下不使用字符。
- lines terminated by 'value'：设置每行结尾的字符，可以为单个或多个字符，默认值为“\n”。如“lines terminated by '?'”表示一行以“?”作为结束标志。

由于 MySQL 数据库版本的变化，在实际过程中往往会发生一些意外错误。对于

MySQL 5.7 来说，数据导出时也需要进行讨论。

例如，使用 select...into outfile 语句将 course 表的内容导出外储文件，系统将表 course 的数据备份在 course.bak 中，默认保存在 data 目录下(C:\ProgramData\MySQL\MySQL Server 5.7\data\)。执行下列语句的结果如下：

```
mysql > use teaching;
      Database changed
mysql > select * from course into outfile 'course.bak';
ERROR 1290 (HY000): The MySQL server is running with the -- secure - file - priv option so it
cannot execute this statement
```

根据错误查找相关资料，发现是 secure-file-priv 会指定文件夹作为导出文件存放的地方，可以先找出这个文件夹。在 MySQL 命令行界面输入以下指令，可以看到如图 11-3 所示的结果。

```
mysql> show variables like '%secure%';
+--------------------------+------------------------------------------------+
| Variable_name            | Value                                          |
+--------------------------+------------------------------------------------+
| require_secure_transport | OFF                                            |
| secure_auth              | ON                                             |
| secure_file_priv         | C:\ProgramData\MySQL\MySQL Server 5.7\Uploads\ |
+--------------------------+------------------------------------------------+
3 rows in set, 1 warning (0.00 sec)
```

图 11-3　查找--secure-file-priv 路径

根据查询标注出来的即是正确的文件路径，将导出文件放在该目录下即可。对于上述 MySQL 命令行指令，修改并执行如下：

```
mysql > select * from course into outfile
    -> 'C:/programdata/MySQL/MySQL Server 5.7/uploads/course.bak';
       Query OK, 13 rows affected (0.00 sec)
```

这样就可以将表 course 的数据导出到对应文件夹下，成功后可以在对应文件夹下看到导出备份文件中的数据。

【例 11-8】 备份一个单独的表 student。

代码和运行结果如下：

```
mysql > select * into outfile
    -> 'c:/ProgramData/MySQL/MySQL Server5.7/Uploads/student.txt'
    -> from student;
       Query OK, 11 rows affected (0.00 sec)
```

【例 11-9】 将表 student 数据分别备份成.xls 和.xml 格式。

代码和运行结果如下：

```
mysql > select * into outfile
    -> 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/student.xls'
    -> from student;
       Query OK, 11 rows affected (0.00 sec)
mysql > select * into outfile
    -> 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/ student.xml'
    -> from student;
```

```
Query OK, 11 rows affected (0.00 sec)
```

说明：

（1）导出的数据可以按照如.txt、.xls、.doc、.xml 等规定格式，通常是.txt 文件。导出的是纯数据，不存在建表信息，也可以直接导入到另外一个同数据库的不同表中，当然表结构要相同。

（2）备份一个庞大的数据库，输出文件也将很庞大，难于管理，可以把数据表进行单独或者几个表一起备份，将备份文件分成较小、更易于管理的文件。

【例 11-10】 使用 select…into outfile 命令将 teaching 数据库中的 score 表中的记录导出到文本文件，使用 fields 选项和 lines 选项，要求字段之间使用逗号“,”间隔，所有字段值用双引号括起来，定义转义字符为单引号“\'”。

代码和运行结果如下：

```
mysql > select * from score into outfile
    -> 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/score.txt'
    -> fields
    ->    terminated by ','
    ->    enclosed by '\"'
    ->    escaped by '\''
    -> lines
    ->    terminated by '\r\n';
Query OK, 27 rows affected (0.03 sec)
```

11.5.2 用 MySQL 命令导出文本文件



导出文本文件

MySQL 命令可用来登录 MySQL 服务器和还原备份文件，也可以导出文本文件。

MySQL 命令导出文本文件的基本语法形式如下：

```
mysql -u root -pPassword -e| -- execute = "select statement" databasename > C:/name.txt ;
```

说明：

password 表示 root 用户的密码；使用-e|--execute＝选项就可以执行 SQL 语句；“select 语句”用来查询记录；“c:/name.txt”表示导出文件的路径。

【例 11-11】 使用 MySQL 命令将 teaching 数据库中的 teacher 表中的记录导出到文本文件。

代码和运行结果如下：

```
mysql -uroot -p -- execute = "select * from teacher;" teaching > d:/bak/teach.txt
Enter password: ******
```

或

```
mysql -u root -p -e "select * from teacher;" teaching > d:/bak/teatxt.txt
Enter password: ******
```

语句执行完毕后，会在 d 盘的 bak 文件夹中生成文件 teach.txt 或 teatxt.txt me。

11.5.3 用 load data infile 方式导入文本文件

MySQL 中，可以使用 load data infile 命令将文本文件导入到 MySQL 数据库中。

load data 命令的基本语法形式如下：

```
load data [low_prioriy|concurrent] infile 'filename.txt'[replace|ignore] into table tablename
[options] [ ignore number lines ] [(columnname [ | UserVariables ], ...)] [ set columnname =
expression, ...)] ;
```

说明：

(1) low_prioriy | concurrent：若指定 low_prioriy，则延迟语句的执行。

(2) filename.txt：该文件中保存了待存入数据库的数据行，由 select...into outfile 语句命令导出产生。

(3) tablename：该表在数据库中必须存在，表结构必须与导入文件的数据行一致。

(4) replace | ignore：如果指定了 replace，则当文件中出现与原有行相同的唯一关键字值时，输入行会替换原有行。

(5) [options]参数选项：与 select...into outfile 语句中的 options 类似。

- fields terminated by 'value'
- fields [optionally] enclosed by 'value'
- fields escaped by 'value'
- lines starting by 'value'
- lines terminated by 'value'

(6) ignore number lines：这个选项可以用于忽略文件的前几行。number 表示忽略的行数。

(7) [(columnname[|UserVariables], ...)]：如果需要载入一个表的部分列或文件中字段值顺序与表中列的顺序不同，就必须指定一个列清单。

(8) [set columnname = expression, ...)]：set 子句可以在导入数据时修改表中列的值。

【例 11-12】 恢复 student 表数据。尝试用 delete 删除 student 表的某些数据或全部数据。

代码和运行结果如下：

```
mysql > delete from student;
        Query OK, 11 rows affected (0.09 sec)
mysql > load data infile
    -> 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/ student.xls'
    -> into table student;
        Query OK, 11 rows affected (0.12 sec)
        Records: 11 Deleted: 0 Skipped: 0 Warnings: 0
```

【例 11-13】 用备份好的 student.txt 文件恢复 student 表数据。为避免主键冲突，要用 replace into table 直接将数据进行替换来恢复数据。

代码和运行结果如下：

```
mysql > load data infile
    -> 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/student.txt'
    -> replace into table student ;
       Query OK, 22 rows affected (0.43 sec)
       Records: 11 Deleted: 11 Skipped: 0 Warnings: 0
```

说明：

(1) 如果表结构破坏，不能用 load data infile 恢复数据，要先恢复表结构。

(2) 如果只是删除了部分数据，例如，删除了某位学生的记录，大部分记录仍在。

(3) 可用"select * from student;"语句查看恢复情况。

【例 11-14】 使用 load data infile 命令将'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/score.txt'文件中的数据导入到 teaching 数据库中的 score 表中，使用 fields 选项和 lines 选项，要求字段之间使用逗号","间隔，所有字段值用双引号括起来，定义转义字符为单引号"\'"。

代码和运行结果如下：

```
mysql > delete from score;
       Query OK, 27 row affected (0.03 sec)
mysql > load data infile
    -> 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/score.txt'
    -> into table score
    ->     fields
    ->         terminated by ','
    ->         enclosed by '\"'
    ->         escaped by '\''
    ->     lines
    ->         terminated by '\r\n';
       Query OK, 27 rows affected (0.07 sec)
       Records: 27 Deleted: 0 Skipped: 0 Warnings: 0
```

11.6 小　　结

本章介绍了备份数据库、还原数据库、数据库迁移、导出表和导入表的内容。在实际应用中，通常使用 mysqldump 命令备份数据库，使用 MySQL 命令还原数据库。数据库迁移需要考虑数据库的兼容性问题，最好是在相同版本的 MySQL 数据库之间迁移。学习本章之后，重点掌握如下内容：

- 数据库备份的原因的分类。
- 数据库恢复时需要注意的问题。
- 数据库备份的基本格式和基本操作。
- 数据库还原的基本格式和基本操作。
- 表数据的导入与导出基本操作。

习　题　11

1. 选择题

(1) 在数据库系统生命周期中可能发生的灾难不包括________。

A. 系统故障　　B. 事务故障　　C. 掉电故障　　D. 介质故障

(2) 按备份时服务器是否在线划分不包括________备份。

A. 热备份　　B. 完全备份　　C. 冷备份　　D. 温备份

(3) 还原数据库时,首先要进行________操作。

A. 创建数据表备份　　B. 创建完整数据库备份

C. 创建冷设备　　D. 删除最近事务日志备份

(4) 创建数据库文件或文件组备份时,首先要进行________操作。

A. 创建事务日志　　B. 创建完整数据库备份

C. 创建温备份　　D. 删除差异备份

(5) 下面故障发生时,________需要数据库管理员进行手工操作恢复。

A. 停电　　B. 误删表数据　　C. 死锁　　D. 操作系统错误

2. 思考题

(1) 为什么在 MySQL 中需要进行数据库的备份与恢复操作?

(2) MySQL 数据库备份与恢复的常用方法有哪些?

(3) 使用直接复制方法实现数据库备份与恢复时,需要注意哪些事项?

(4) 进行数据库还原应该注意哪些问题?

(5) 备份数据库的时机如何选择?

3. 上机练习题(本题利用 teaching 数据库进行操作)

(1) 使用 mysqldump 命令备份数据库 teaching 中的所有表。

(2) 使用 source 命令将备份文件 teachingbak.sql 恢复到数据库中。

(3) 使用 mysqldump 命令备份数据库中的 score 表。

(4) 删除 score 表的数据,用 source 命令恢复。

(5) 使用 MySQL 命令将 teaching 数据库中的 course 表中的记录导出到文本文件。

(6) 用备份好的 teach.txt 文件恢复 course 表数据。为避免主键冲突,要用 replace into table 直接将数据进行替换来恢复数据。

第 12 章　MySQL 性能优化

优化性能是通过某些高效的方法提高 MySQL 数据库的性能，其目的是为了使 MySQL 数据库运行速度更快、占用的磁盘空间更小。性能优化包括优化查询速度、优化更新速度、优化 MySQL 服务器等。优化 MySQL 数据库是数据库管理员的必备技能。

在实际工作中，数据的查询优化可以有效地提高 MySQL 数据库的性能。一个成功的数据库应用系统的开发，在查询优化方面一定会付出资源。对查询优化的处理，不仅会影响到数据库的工作效率，而且会给社会带来较高的效益。

本章将学习优化 MySQL 服务器、优化数据表、优化查询的方法和技巧。

12.1　优化 MySQL 服务器

MySQL 数据库的用户和数据的量达不到一定的规模，MySQL 数据库的性能的好坏很难判断。当有大量用户进行长时间频繁操作的运行，数据库的性能才能体现出来。当大量用户同时连接 MySQL 数据库进行查询、插入和更新操作时，如果数据库的性能很差，就很可能无法承受如此多用户同时操作，会出现数据库系统瘫痪的状况。

优化服务器是 MySQL 数据库管理的重要方法。优化 MySQL 服务器可以从两个方面来理解。一个是从硬件方面来进行优化，另一个是从 MySQL 服务的参数进行优化。

12.1.1　优化服务器硬件

服务器的硬件性能直接决定着 MySQL 数据库的性能。硬件的性能瓶颈，直接决定 MySQL 数据库的运行速度和效率。例如，增加内存和提高硬盘的读写速度，这能够提高 MySQL 数据库的查询、更新的速度。

硬件技术的成熟使得硬件的价格也随之降低。一般的 PC 都配置 4GB 的内存，一些 PC 配置 8GB 的内存，甚至有 16GB 的内存。因为内存的读写速度比硬盘的读写速度快，可以在内存中为 MySQL 设置更多的缓冲区，这样可以提高 MySQL 访问的速度。如果将查询频率很高的记录存储在内存中，那么查询速度就会很快。

对于支持 InnoDB 存储引擎的表来说，如果条件允许，可以将内存提高到 8GB，且选用 my-innodb-heavy-8G.ini 作为 MySQL 数据库的配置文件。MySQL 所在的计算机最好是专用数据库服务器。这样数据库可以完全利用该机器的资源以提高数据的查询速度，优化查询性能。

12.1.2 修改 my.ini 文件

如果 MySQL 数据库需要进行大量的查询操作，那么就需要对查询语句进行优化。对于耗费时间的查询语句进行优化，可以提高整体的查询速度。如果连接 MySQL 数据库用户很多，那么就需要对 MySQL 服务器进行优化。

MySQL 配置文件（my.ini）保存了服务器的配置信息，通过修改 my.ini 文件的配置可以优化服务器，提高性能。

在默认情况下，MySQL 数据库索引的缓冲区大小为 16MB，为得到更好的索引处理性能，可以打开修改 my.ini 文件，重新设置索引的缓冲区大小，例如可以在[MySQLd]后面加上一行代码设定索引缓冲区为 256MB。

```
key_buffer_size = 256M
```

如果 MySQL 服务器的计算机内存为 4GB，则主要的几个参数推荐设置如下：

```
sort_buffer_size = 6M         //查询排序时所能使用的缓冲区大小
read_buffer_size = 4M         //读查询操作所能使用的缓冲区大小
join_buffer_size = 8M         //联合查询操作所能使用的缓冲区大小
query_cache_size = 64M        //查询缓冲区的大小
max_connections = 800         //指定 MySQL 允许的最大连接进程数
```

12.1.3 通过 MySQL 控制台进行性能优化

数据库管理人员可以使用 show status 或 show variables like 语句来查询 MySQL 数据库的性能参数，然后用 set 语句对系统变量进行赋值。

1. 查询主要性能参数

（1）利用 show status 语句查询 MySQL 数据库的性能语法形式如下：

```
show status like 'value';
```

说明：

使用 value 参数时常用的几个统计参数介绍如下。

- Connections：连接 MySQL 服务器的次数。
- Uptime：MySQL 服务器的上线时间。
- Slow_queries：慢查询的次数。
- Com_select：查询操作的次数。
- Com_insert：插入操作的次数。
- Com_update：更新操作的次数。
- Com_delete：删除操作的次数。

例如，如果需要查询次数，可以执行下面的 show status 语句：

```
mysql > show status like 'Com_select';
        +------------+-----+
        | Variable_name  | Value |
        +------------+-----+
```

```
| Com_select      |   23  |
+-------------+-----+
1 row in set (0.00 sec)
```


通过这些参数可以分析 MySQL 数据库的性能参数，根据分析结果，进行相应的性能优化。

（2）利用 show variables like 语句查询 MySQL 数据库的性能语法形式如下：

```
show variables like 'value';
```

说明：

其中，value 参数常用的几个统计参数如下。

- key_buffer_size：表示索引缓存的大小。
- table_cache：表示同时打开的表的个数。
- query_cache_size：表示查询缓冲区的大小。
- Query_cache_type：表示查询缓存区的开启状态。0 表示关闭，1 表示开启。
- Sort_buffer_size：排序缓冲区的大小，这个值越大，排序就越快。
- Innodb_buffer_pool_size：表示 InnoDB 类型的表和索引的最大缓存。这个值越大，查询的速度就会越快。但是，这个值太大了也会影响操作系统的性能。

2. 设置性能指标参数

例如，要设置查询缓冲区的系统变量，可先执行以下命令进行观察。

代码和运行结果如下：

```
mysql > show variables like '%query_cache%';
        +-----------------------+-------+
        | Variable_name                | Value   |
        +-----------------------+-------+
        | have_query_cache             | YES     |
        | query_cache_limit            | 1048576 |
        | query_cache_min_res_unit     | 4096    |
        | query_cache_size             | 0       |
        | query_cache_type             | off     |
        | query_cache_wlock_invalidate | off     |
        +-----------------------+-------+
        6 rows in set, 1 warning (0.17 sec)
```

说明：

（1）Query_cache_type：表示查询缓冲区的开启状态。0 表示关闭，1 表示开启。查询缓冲区主要是为了提高经常执行相同的查询操作的速度，但是，另一方面查询缓冲区也无形中增加了系统的开销，所以有时为减少系统的开销，也可以关闭查询缓冲区。

例如，如果输入如下命令：

```
mysql > use MYSQL;
mysql > set @@Query_cache_type = 0;
```

则所有查询不使用查询缓冲区。但不会导致 MySQL 释放 query_cache_size 配置的缓冲区内存。如果设置 Query_cache_type=1，所有查询都将使用查询缓冲区。

(2) 如果要禁用查询缓存,可以设置 query_cache_size=0,禁用查询缓存后,将没有明显的开销。例如:

```
Mysql > set @@global.query_cache_size = 0;
```

(3) query_cache_limit:表示不要缓存大于该值的结果,默认值是 1048576b(1MB)。如果要设置缓存不大于 64MB(64 * 1024 * 1024=67108864b),可以输入如下命令:

```
mysql > set @@global.query_cache_limit = 67108864;
```

再来查看查询缓冲区系统变量的情况。

代码和运行结果如下:

```
mysql > show variables like '%query_cache%';
        +-----------------------+--------+
        | Variable_name         | Value  |
        +-----------------------+--------+
        | have_query_cache      | YES    |
        | query_cache_limit     | 67108864|
        | query_cache_min_res_unit | 4096 |
        | query_cache_size      | 0      |
        | query_cache_type      | off    |
        | query_cache_wlock_invalidate | off |
        +-----------------------+--------+
        6 rows in set, 1 warning (0.00 sec)
```

执行上述命令可以观察到参数发生的相应改变。

12.2 优化查询

MySQL 作为 Web 数据库,每天要接受来自 Web 的成千上万用户的连接访问。在对数据库频繁操作访问的情况下,数据库的性能好坏越来越成为整个应用的性能瓶颈。

学习使用 explain 语句可以对 select 语句的执行效果进行分析,通过分析提出优化查询的方法;使用 analyze table 语句可以分析表查询效率等。

使用 check 语句检查表,使用 optimize table 语句优化表;学习使用 repair table 语句来修复表的方法。

12.2.1 分析查询语句

分析查询语句在前面内容中都有应用,在 MySQL 中,可以使用 explain 语句和 describe 语句来分析查询语句。

1. explain 语句

应用 explain 关键字分析查询语句,其语法结构如下:

```
explain select statements;
```

说明:

select statements 参数为一般数据库查询命令,如"select * from student;"。



explain 语句

【例 12-1】 使用 explain 语句分析一个查询语句。

代码如下：

```
mysql > use teaching;
        Database changed
mysql > explain select * from course ;
```

运行结果如图 12-1 所示。

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	course	NULL	ALL	NULL	NULL	NULL	NULL	13	100.00	NULL

1 row in set, 1 warning (0.52 sec)

图 12-1 explain 语句的运行结果

说明：

explain 语句输出行的相关信息所代表的意义如下所示：

(1) id：指出在整个查询中 select 的位置。

(2) select_type：表示查询的类型。参数有几项常用的取值，如表 12-1 所示。

表 12-1 select_type 查询类型参数表

参 数 名	作 用
simple	简单 select(不使用 union 和子查询)
primary	表示主查询或者是最外面的 select 语句
union	表示连接查询(union)中的第二个或后面的 select 语句
subquery	子查询中的第一个 select 语句

(3) table：查询的源表名。

(4) partitions：查询的源表是否分区。

(5) type：显示连接使用了哪种连接类别，是否使用索引，是使用 explain 命令分析性能瓶颈的关键项之一。该列中存储很多值，范围从 const 到 all。按照从最佳类型到最坏类型进行排序，system > const > eq_ref > ref > fulltext > ref_or_null > index_merge > unique_subquery > index_subquery > range > index > all 。一般来说，要保证查询至少达到 range 级别，最好能达到 ref，否则就可能会出现性能问题。表 12-2 列出了几项常用的参数取值。

表 12-2 type 列常用参数取值

参 数 名	作 用
system	表示表中只有一条记录
const	表示表中有多条记录，但只从表中查询一条记录
eq_ref	表示多表连接时，后面使用了 unique 或者 primary key
ref	表示多表查询时，后面的表使用了普通索引
unique_subquery	表示子查询中使用了 unique 或者 primary key
index_subquery	表示子查询使用了普通索引
range	表示查询语句给出了查询范围
index	表示对表中的索引进行了完整的扫描，比 all 快一些
all	表示对表中数据进行全扫描

(6) possible_keys：指出为了提高查找速度，查询在 MySQL 中可能用到哪个索引。如果该列是 null，则没有相关的索引。

(7) key：显示查询实际使用的键(索引)。

(8) key_len：显示使用的索引字段的长度。

(9) ref：显示使用哪个列或常数与索引一起来查询记录。

(10) rows：显示执行查询时必须检查的行数。

(11) filtered：筛选的结果。

(12) extra：包含解决查询的附加信息。想要让查询尽可能地快，那么就应该注意 extra 字段的值为 usingfilesort 和 using temporary 的情况，具体作用如表 12-3 所示。

表 12-3　extra 常用参数取值

参　数　名	作　　用
distinct	一旦找到了与查询条件匹配的第一条记录后，就不再搜索其他记录
not exists	MySQL 优化了 left join，一旦它找到了匹配 left join 标准的行，就不再搜索更多的记录
range checked for each record (index map：#)	没找到合适的可用的索引。对于前一个表的每一个行连接，它会做一个检验以决定该使用哪个索引(如果有的话)，并且使用这个索引来从表里取得记录。这个过程不会很快，但总比没有任何索引时做表连接要快些
using filesort	MySQL 需要进行额外的步骤以排好的顺序取得记录，查询需要优化
using index	字段的信息直接从索引树中的信息取得，而不再去扫描实际的记录。这种策略用于查询时的字段是一个独立索引的一部分
using temporary	MySQL 需要创建一个临时表来存储结果，这通常发生在查询时包含了 order by 和 group by 子句，以不同的方式列出了各种字段，查询需要优化
using where	使用了 where 从句来限制哪些行将与下一张表匹配或者是返回给用户

2. describe 语句

在 MySQL 中应用 describe 语句来分析查询语句，其使用方法与 explain 语法是相同的，这两者的分析结果也大体相同。describe 可以缩写成 desc 命令。

describe 的语法结构如下：

```
describe select statements;
```

【例 12-2】 利用 describe 命令分析查询语句。

代码如下：

```
mysql > describe select * from student;
```

运行结果如图 12-2 所示。

```
+----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra |
+----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+-------+
|  1 | SIMPLE      | student | NULL       | ALL  | NULL          | NULL | NULL    | NULL |   11 |   100.00 | NULL  |
+----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+-------+
1 row in set, 1 warning (0.14 sec)
```

图 12-2　describe 语句的运行结果

说明：

将例 12-2 与例 12-1 对比，可以清楚地看出，其运行结果基本相同。

12.2.2 索引对查询速度的影响



索引对查询速度的影响

在查询过程中使用索引，势必会提高数据库的查询效率，应用索引来查询数据库中的内容，可以减少查询的记录数，从而达到查询优化的目的。

下面将通过对使用索引和不使用索引进行对比，来分析查询的优化情况。

【例 12-3】 分析索引对查询速度的影响，在未使用索引时的查询情况。

代码如下：

```
mysql > explain select * from student where sname = '崔依歌';
```

运行结果如图 12-3 所示。

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	student	NULL	ALL	NULL	NULL	NULL	NULL	11	10.00	Using where

1 row in set, 1 warning (0.02 sec)

图 12-3 未使用索引的查询情况

图 12-3 的结果中，只是使用了 where 从句的一个简单查询，没有使用索引进行查询，type 为 all 表示要对表进行全扫描。表格字段 rows 下为 11，说明在执行查询的过程中，student 表中存在的 11 条数据都被查询了一遍。可以想象，在数据存储量小的时候，查询不会有太大影响，当数据库中存储海量的数据时，为搜索一条数据而遍历整个数据表中的所有记录，将会耗费很多时间。

如果在 sname 字段上建立一个名为 idx_sname 的索引，然后应用 explain 关键字分析执行情况，就可以观察到索引的作用。

代码如下：

```
mysql > create index idx_sname on student(sname);
        Query OK, 0 rows affected (2.34 sec)
        Records: 0 Duplicates: 0 Warnings: 0
mysql > explain select * from student where sname = '崔依歌';
```

运行结果如图 12-4 所示。

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	student	NULL	ref	idx_sname	idx_sname	16	const	1	100.00	NULL

1 row in set, 1 warning (0.04 sec)

图 12-4 使用索引的查询情况

如图 12-4 所示，由于创建的索引使访问的行数由 11 行减少到 1 行。type 级别已经上升至 ref，明显地提高了查询性能。其实际过程是，因为创建了 idx_sname 的索引，查询姓名“崔依歌”时，先在索引文件中查找键值，再通过其 id 号到数据表中查找相关记录。如果数据量较小，顺序查询比索引查询要快。当数据量较大时，索引查询效率高。

由此可见，在查询操作中，使用索引不但会自动优化查询效率，同时也会降低服务器的开销。

一般情况下，使用索引可以提高查询的速度，但如果 MySQL 语句使用不恰当的话，索引将无法发挥它应有的作用。如果在一个表中创建了多列的复合索引，只有查询条件中使用了这些字段的第一个字段时，索引才会使用。

12.2.3 使用索引优化查询

在 MySQL 中，可以通过索引提高查询的速度。为了更充分地发挥索引的作用，在应用索引查询时，可以通过关键字或其他方式来对查询进行优化处理。



应用 like 优化索引查询

1. 应用 like 关键字优化索引查询

【例 12-4】 利用 explain 语句执行查询命令，应用 like 关键字，且匹配字符串中含有的百分号"%"符号。

代码和运行结果如下：

```
mysql> explain select * from student where sname like '赵%'\G
        *************************** 1. row ***************************
                   id: 1
          select_type: SIMPLE
                table: student
           partitions: NULL
                 type: range
        possible_keys: idx_sname
                  key: idx_sname
              key_len: 16
                  ref: NULL
                 rows: 2
             filtered: 100.00
                Extra: Using index condition
          1 row in set, 1 warning (0.06 sec)
```

再执行下面的类似查询：

```
mysql> explain select * from student where sname like '%江'\G
        *************************** 1. row ***************************
                   id: 1
          select_type: SIMPLE
                table: student
           partitions: NULL
                 type: ALL
        possible_keys: NULL
                  key: NULL
              key_len: NULL
                  ref: NULL
                 rows: 11
             filtered: 11.11
                Extra: Using where
          1 row in set, 1 warning (0.00 sec)
```

说明：

从上面的两个运行结果中可以看出，sname 列使用了索引，都与 like 关键字进行匹配。如果匹配字符（%或_）在字符串的后面，索引在其中起作用，如第一种情况，type 值为 range 级，因为有两条符合条件的记录，所以 rows 参数值为 2，检查的行数只有 2 行。

对于第 2 种情况，匹配字符（%或_）在字符串的前面，索引将不起作用，type 值为 all 级，即对表进行全扫描，检查的行数为 11 行，虽然符合条件的记录行为 3 行。

由此可以知道，使用 like 关键字和通配符的做法虽然简单、易懂，但却也是以牺牲系统性能为代价的。

2. 查询语句中使用 or 关键字

在 MySQL 中，查询语句只有包含 or 关键字时，要求查询的两个字段必须同为索引，如果搜索的条件中，有一个字段不为索引，则在查询中不会应用索引进行查询。



应用 or 优化查询

【例 12-5】 通过 explain 来分析应用 or 关键字查询索引的命令。

代码和运行结果如下：

```
mysql > explain select * from student where sname = '赵 % ' or phone = '132 % '\G
    ************************** 1. row ***************************
               id: 1
      select_type: SIMPLE
            table: student
       partitions: NULL
             type: index_merge
    possible_keys: phone_index, idx_sname
              key: idx_sname, phone_index
          key_len: 16,26
              ref: NULL
             rows: 2
         filtered: 100.00
            Extra: Using union( idx_sname, phone_index); Using where
      1 row in set, 1 warning (0.00 sec)
```

再执行下面的类似查询：

```
mysql > explain select * from student where sname = '赵 % ' or sex = '男 '\G
    ************************** 1. row ***************************
               id: 1
      select_type: SIMPLE
            table: student
       partitions: NULL
             type: ALL
    possible_keys: idx_sname
              key: NULL
          key_len: NULL
              ref: NULL
             rows: 11
         filtered: 54.55
            Extra: Using where
      1 row in set, 1 warning (0.02 sec)
```

从运行结果中可以看出，若两个字段均为索引，故查询被优化，type 的值为 index_merge。而后一种情况，由于 sex 字段没有被索引，则查询速度不会被优化。type 值为 all，表示进行了全扫描，rows 值为 11。

3. 查询语句中使用多列索引



多列索引的优化查询

多列索引在表的多个字段上创建一个索引，只有查询条件中使用了这些字段中的第一个字段时，索引才会被正常使用。

【例 12-6】 通过 explain 来分析应用多列索引的命令，score 表中有索引 studentno＋ courseno，分别用这两个字段进行查询分析。



代码和运行结果如下：

```
mysql> explain select * from score where studentno = '18%'\G
        *************************** 1. row ***************************
                   id: 1
          select_type: SIMPLE
                table: score
           partitions: NULL
                 type: ref
        possible_keys: PRIMARY,sc_index
                  key: PRIMARY
              key_len: 22
                  ref: const
                 rows: 1
             filtered: 100.00
                Extra: NULL
          1 row in set, 1 warning (0.22 sec)
```

再执行下面的类似查询：

```
mysql> explain select * from score where courseno = 'c05%'\G
        *************************** 1. row ***************************
                   id: 1
          select_type: SIMPLE
                table: score
           partitions: NULL
                 type: ALL
        possible_keys: NULL
                  key: NULL
              key_len: NULL
                  ref: NULL
                 rows: 30
             filtered: 10.00
                Extra: Using where
          1 row in set, 1 warning (0.00 sec)
```

说明：

在应用 courseno 字段时，索引不能被正常使用，进行的是全表扫描，这就是说索引并未在 MySQL 优化中起到任何作用。而必须使用字段 studentno 时，索引才可以被正常使用。

4. 在索引字段上使用函数操作

在建有索引的字段上尽量不要使用函数进行操作，否则会降低查询速度。例如，在一个

date 类型的字段上使用 year()函数时，将会使索引不能发挥应有的作用。

【例 12-7】 在 student 表的 birthdate 字段已建立了索引 idx_birth，进行的两个查询结果对比一下。

分析使用函数的查询优化

代码和运行结果如下：

```
mysql > create index idx_birth on student(birthdate);
        Query OK, 0 rows affected (0.63 sec)
        Records: 0 Duplicates: 0 Warnings: 0
mysql > explain select sname from student where year()(birthdate)> '2000'\G
         *************************** 1. row ***************************
                   id: 1
          select_type: SIMPLE
                table: student
           partitions: NULL
                 type: ALL
        possible_keys: NULL
                  key: NULL
              key_len: NULL
                  ref: NULL
                 rows: 11
             filtered: 100.00
                Extra: Using where
          1 row in set, 1 warning (0.00 sec)
mysql > explain select sname from student where birthdate >'2000 - 12 - 31'\G
         *************************** 1. row ***************************
                   id: 1
          select_type: SIMPLE
                table: student
           partitions: NULL
                 type: ALL
        possible_keys: idx_birth
                  key: NULL
              key_len: NULL
                  ref: NULL
                 rows: 11
             filtered: 81.82
                Extra: Using where
          1 row in set, 1 warning (0.00 sec)
```

说明：

在一个 date 类型的字段上使用 year()函数时，将会使索引不能发挥应有的作用。possible_keys 值为 null，而没有使用 year()函数时，possible_keys 值为 idx_birth，证明查询过程中利用了索引 idx_birth。但是，由于数据量较小，在指定查询计划时，索引使用的优势显示不明显。

12.2.4 优化多表查询

优化多表查询

在 MySQL 中，很多查询中需要使用子查询。子查询可以使查询语句很灵活，但子查询的执行效率不高。进行子查询时，MySQL 需要为内层

查询语句的查询结果建立一个临时表。然后外层查询语句在临时表中查询记录。查询完毕后，MySQL 需要撤销这些临时表。因此，子查询的速度会受到一定的影响。如果查询的数据量比较大，这种影响就会随之增大。在 MySQL 中可以使用连接查询来替代子查询。连接查询不需要建立临时表，其速度比子查询要快。

用户可以通过连接来实现多表查询，在查询过程中，用户将表中的一个或多个共同字段进行连接，定义查询条件，返回统一的查询结果。这通常用来建立数据管理系统的数据表之间的关系。在多表查询中，可以应用子查询来优化多表查询，即在 select 语句中嵌套其他 select 语句。采用子查询优化多表查询的好处有很多，其中，可以将分步查询的结果整合成一个查询，这样就不需要再执行多个单独查询，从而提高了多表查询的效率。

【例 12-8】 通过一个实例来说明如何优化多表查询，查看子查询方式和表连接方式的查询分析参数。先执行两个查询语句，再执行查询分析。

代码和运行结果如下：

```
mysql > select sname ,phone from student where studentno
          in (select studentno from score where final > 98)\G
        *************************** 1. row ***************************
        sname: 敬横江
        phone: 15678945623
        *************************** 2. row ***************************
        sname: 赵既白
        phone: 13175689345
        2 rows in set (0.00 sec)
mysql > select sname,phone from student,score
          where student. studentno  = score.studentno and final > 98 \G
        *************************** 1. row ***************************
        sname: 敬横江
        phone: 15678945623
        *************************** 2. row ***************************
        sname: 赵既白
        phone: 13175689345
        2 rows in set (0.00 sec)
#子查询方式分析
mysql > explain select sname ,phone from student where studentno
          in (select studentno from score where final > 98)\G
        *************************** 1. row ***************************
                   id: 1
          select_type: SIMPLE
                table: student
           partitions: NULL
                 type: ALL
        possible_keys: PRIMARY
                  key: NULL
              key_len: NULL
                  ref: NULL
                 rows: 11
             filtered: 100.00
                Extra: Using where
```

```
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: <subquery2>
   partitions: NULL
         type: eq_ref
possible_keys: <auto_key>
          key: <auto_key>
      key_len: 22
          ref: teaching.student.studentno
         rows: 1
     filtered: 100.00
        Extra: NULL
*************************** 3. row ***************************
           id: 2
  select_type: MATERIALIZED
        table: score
   partitions: NULL
         type: ALL
possible_keys: PRIMARY,sc_index
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 30
     filtered: 33.33
        Extra: Using where
  3 rows in set, 1 warning (0.00 sec)
#表连接方式分析
mysql> explain select sname, phone from student, score where student. studentno = score.
studentno and final > 98 \G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: score
   partitions: NULL
         type: ALL
possible_keys: PRIMARY,sc_index
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 30
     filtered: 33.33
        Extra: Using where
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: student
   partitions: NULL
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
```

```
      key_len: 22
          ref: teaching.score.studentno
         rows: 1
     filtered: 100.00
        Extra: NULL
2 rows in set, 1 warning (0.00 sec)
```

说明：

(1) 从运行结果中可以看出，虽然查询的结果是一样的，但效率却有差别。子查询方式对两个表进行的是全表扫描，子查询本身生成的临时表< subquery2 >也需要扫描。而表连接方式，则仅对 score 表进行全表扫描，而基于 student 表的分析，type 的值为 eq_ref。表明该语句已经将算法进行优化，从而提高了数据表的查询效率，实现了查询优化的效果。

(2) 使用子查询可以一次性地完成很多逻辑上需要多个步骤才能完成的 SQL 操作，同时也可以避免事务或者表锁死，并且写起来也很容易。但是 MySQL 在执行带有子查询的查询时，需要先为内层子查询语句的查询结果建立一个临时表，然后外层查询语句在临时表中查询记录，查询完毕后再撤销这些临时表。

(3) 子查询的速度会受到一定的影响，特别是查询的数据量比较大时，这种影响就会随之增大。因此，尽量使用连接查询来代替子查询，连接查询不需要建立临时表，其速度比子查询要快。

12.3 优化数据库结构

数据表结构是否合理，需要考虑是否存在冗余、对表的查询和更新的速度、表中字段的数据类型是否合理等多方面的内容。

12.3.1 优化表结构

根据数据库表中数据的使用频率，可以视具体情况对表结构进行适当的增减修改，以此提高相关查询的效率。

1. 将字段很多的表分解成多个表

有些表在设计时设置了很多的字段。这个表中有些字段的使用频率很低。当这个表的数据量很大时，查询数据的速度就会很慢。对于这种字段特别多且有些字段的使用频率很低的表，可以将其分解成多个表。

例如，学生表 student 的字段中，entrance 字段中存储着学生的入学成绩信息，很少使用。可以分解出另外一个表，将这个表取名为 stu_entrance。表中存储两个字段，分别为 studentno 和 entrance。

如果需要查询某个学生的入学成绩信息，可以用学号 studentno 来查询。如果需要将学生的学籍信息与备注信息同时显示，可以将 student 表和 stu_entrance 表进行连接查询，通过这种分解，可以提高 student 表的查询效率。

2. 增加冗余字段

有时，在建立表的时候有意识地增加冗余字段，减少连接查询操作，提高性能。

例如，课程的信息存储在 course 表中，成绩信息存储在 score 表中，两表通过课程编号

courseno 建立关联。对于要查询选修某门课(如数据库编程)的学生,必须从 course 表中查找课程名称所对应的课程编号 courseno,然后根据这个编号去 score 表中查找该课程成绩。

为减少查询时由于建立连接查询浪费的时间,可以在 score 表中增加一个冗余字段 cname,该字段用来存储课程的名称。这样就不用每次都进行连接操作了。

3. 合理设置表的数据类型和属性

(1) 选取适用的字段类型。表中字段的宽度设得尽可能小。

例如,在定义地址字段时,一般使用 char 或 varchar。考虑到一般情况下地址字段的长度是 10 个字符左右,没必要设置 char(255),尽量减少不必要的数据库的空间损耗。如果不需要记录时间,使用 date 要比 datetime 好得多。

(2) 使用 enum 而不是 varchar 或 char。对诸如"省份""性别""爱好""民族"或"部门"等字段,可以选择 enum 数据类型。一方面由于这样的字段取值是有限而且固定的,另一方面,MySQL 把 enum 类型当作数值型数据来处理,而数值型数据处理起来的速度要比文本类型快得多。

(3) 为每张表设置一个 id。为每张数据表都设置 id 作为其主键,而且最好的是一个 int 型的主键,并设置自动增量(auto_increment)。

(4) 尽量避免定义 null。一个提高效率的方法是在可能的情况下,尽量把字段设置为 not null,这样在将来执行查询的时候,数据库不用去比较 null 值。

12.3.2 增加中间表

若有某些查询经常涉及多表中的几个字段,就需要进行多表连接。经常进行连接查询,会降低 MySQL 数据的查询速度。此时可以视情况将这些字段建立一个中间表,并将原来那几个表的数据插入到中间表中,就可以使用中间表来进行查询和统计了。

【例 12-9】 利用 student 表、course 表和 score 表的结构,创建中间表 stu_score,包含实际中经常要查询的学生的学号、姓名、课程名和成绩信息。

代码和运行结果如下:

```
#查看 student 表、course 表和 score 表的结构:
mysql> describe student;
        +---------+-------------+----+----+------+-----+
        | Field   | Type        |Null| Key|Default|Extra|
        +---------+-------------+----+----+------+-----+
        | studentno| char(11)   | NO | PRI| NULL |     |
        | sname   | char(8)     | NO | MUL| NULL |     |
        | sex     | enum('男','女')| YES|   | 男   |     |
        | birthdate| date       | NO | MUL| NULL |     |
        | entrance| int(3)      | YES|    | NULL |     |
        | phone   | varchar(12) | NO | UNI| NULL |     |
        | Email   | varchar(20) | NO |    | NULL |     |
        +---------+-------------+----+----+------+-----+
        7 rows in set (0.03 sec)
mysql> desc course;
        +---------+----------------+----+----+------+-----+
        | Field   | Type           |Null| Key|Default|Extra|
```

```
+---------+----------------+----+----+------+-----+
| courseno | char(6)          | NO  | PRI | NULL |      |
| cname    | char(6)          | NO  | UNI | NULL |      |
| type     | enum('必修','选修') | YES |     | 必修  |      |
| period   | int(2)           | NO  |     | NULL |      |
| exp      | int(2)           | NO  |     | NULL |      |
| term     | int(2)           | NO  |     | NULL |      |
+---------+----------------+----+----+--------+---+
6 rows in set (0.08 sec)
mysql > desc score;
+---------+---------+----+----+-------+-----+
| Field     | Type       | Null | Key | Default | Extra |
+---------+---------+------+----+-----+-----+
| studentno | char(11)   | NO   | PRI | NULL    |       |
| courseno  | char(6)    | NO   | PRI | NULL    |       |
| daily     | float(3,1) | YES  |     | 0.0     |       |
| final     | float(3,1) | YES  |     | 0.0     |       |
+---------+---------+----+----+-------+-----+
4 rows in set (0.00 sec)
#创建中间表 stu_score
mysql > create table stu_score as
    -> select student.studentno,sname,cname,daily, final
    -> from student, course ,score
    -> where student.studentno = score.studentno
    ->  and score.courseno = course.courseno;
   Query OK, 22 rows affected (0.20 sec)
   Records: 22 Duplicates: 0 Warnings: 0
```

说明：

创建 stu_score 表以后，可以直接在表中查询学生的学号、姓名、课程名和成绩信息，省去了每次查询时进行表连接。这样就节省了查询时间，提高了 MySQL 数据库的查询性能。

【例 12-10】 统计各科课程的总评成绩的平均分，直接利用 stu_score 查询。

代码和运行结果如下：

```
mysql > select cname, avg(daily * 0.2 + final * 0.8) avg
    -> from stu_score group by cname;
+---------+--------+
| cname    | avg      |
+---------+--------+
| C语言    | 83.26667 |
| 高等数学 | 79.90000 |
| 会计软件 | 87.60000 |
| 机械设计 | 92.60000 |
| 机械制图 | 86.30000 |
| 经济法   | 93.30000 |
+---------+--------+
6 rows in set (0.00 sec)
```

12.3.3 优化插入记录的速度

如果 MySQL 数据表中创建的索引比较多，当需要对表进行插入记录的操作时，就会不

断地刷新索引，自动排序数据。如果插入大量数据，索引、唯一性校验都会影响到插入记录的速度。优化插入记录的速度可以从以下几个方面进行处理。

(1) 禁用索引。为了解决插入记录时，排序过程会降低插入记录速度的情况，在插入记录之前可以先禁用索引。等到记录都插入完毕后再开启索引。

禁用索引和重新开启索引的命令格式如下：

```
alter table tablename disable keys
alter table tablename enable keys;
```

对于新创建的表，可以先不创建索引。等到记录都导入以后再创建索引。这样可以提高导入数据的速度。

(2) 禁用唯一性检查。插入数据时，MySQL 会对插入的记录进行校验。这种校验也会降低插入记录的速度。可以在插入记录之前禁用唯一性检查。等到记录插入完毕后再开启。

禁用唯一性检查和重新开启唯一性检查的命令如下：

```
set unique_checkS = 0;
set unique_checkS = 1;
```

(3) 采用 insert 语句的优选方式。向数据表中插入多条记录时，有两种插入语句的格式。第 1 种是一个 insert 语句只插入一条记录，执行多个 insert 语句来插入多条记录。第 2 种是一个 insert 语句插入多条记录。第 2 种方式减少了与数据库之间的连接等操作，其速度比第 1 种方式要快。

在实际操作过程中，若有大量数据需要插入时，建议使用一个 insert 语句插入多条记录的方式。如果能用 load data infile 语句，就尽量用 load data infile 语句。因为 load data infile 语句导入数据的速度比 insert 语句的速度要快。加载数据时要采用批量加载，尽量减少 MySQL 服务器对索引的刷新频率。

12.3.4 分析表、检查表和优化表

分析表的主要作用是分析关键字的分布。检查表的主要作用是检查表是否存在错误。优化表的主要作用是消除删除或者更新造成的空间浪费。

1. 利用 analyze 语句分析表

MySQL 中使用 analyze table 语句来分析表。analyze table 语句能够分析 InnoDB 和 MyISAM 类型的表。

MySQL 的 optimizer(优化元件)在优化 SQL 语句时，首先需要收集相关信息。其中如果表的 cardinality(散列程度，表示某个索引对应的列包含多少个不同的值)大大少于数据的实际散列程度，那么索引就基本失效了。

analyze 语句的基本语法如下：

```
analyze table tablename1[,tablename2...];
```

使用 analyze table 分析表的过程中，数据库系统会对表加一个只读锁。在分析期间，只能读取表中的记录，不能更新和插入记录。

【例 12-11】 分析 teacher 表的运行情况，先使用 show index 语句来查看索引的散列程

度，然后可以使用 analyze table 进行修复。

代码和运行结果如下：

```
mysql > show index from teacher\G;
        ************************* 1. row *************************
                Table: teacher
           Non_unique: 0
             Key_name: PRIMARY
         Seq_in_index: 1
          Column_name: teacherno
            Collation: A
          Cardinality: 9
             Sub_part: NULL
               Packed: NULL
                 Null:
           Index_type: BTREE
              Comment:
        Index_comment:
        1 row in set (0.05 sec)
mysql > analyze table teacher ;
        +---------------+--------+---------+---------+
        | Table         | Op     | Msg_type | Msg_text |
        +---------------+--------+---------+---------+
        | teaching.teacher | analyze | status  | OK      |
        +---------------+--------+---------+---------+
        1 row in set (0.06 sec)
```

说明：

(1) 从例 12-11 的结果可以看到，索引字段是 teacherno，teacher 表的 cardinality 的值为 9。teacher 表的 teacherno 数量为 9，说明索引是有效的。如果 teacher 表的 teacherno 数量远远大于 cardinality 的值，则索引是无效的。

(2) 例 12-11 的结果显示了 4 列信息，检查表和优化表之后也会出现这 4 列信息。其基本含义如下：

- Table：表示表的名称。
- Op：表示执行的操作。analyze 表示进行分析操作，check 表示进行检查查找，optimize 表示进行优化操作。
- Msg_type：表示信息类型，其显示的值通常是状态、警告、错误和信息这四者之一。
- Msg_text：显示信息。

2. 使用 check 语句来检查表

MySQL 中的 check table 语句能够检查 InnoDB 和 MyISAM 类型的表是否存在错误。数据库经常可能遇到的错误，如数据写入磁盘时发生错误，或是索引没有同步更新，或是数据库未关闭 MySQL 就停止了，数据库就可能发生错误。此时，可以使用 Check Table 来检查表是否有错误。该语句还可以检查视图是否存在错误。

该语句的基本语法如下：

```
check table tablename1[,tablename2,...][option];
```

其中，option 有 5 个参数，分别是 quick、fast、changed、medium 和 extended。这 5 个参

数的执行效率依次降低。option 选项只对 MyISAM 类型的表有效，对 InnoDB 类型的表无效。check table 语句在执行过程中也会给表加上只读锁。

例如，检查 student 表的运行情况，语句如下：

```
mysql > Check Table student;
```

3. 使用 optimize 语句来优化表

MySQL 中的 optimize table 语句对 InnoDB 和 MyISAM 类型的表都有效。当表上的数据行被删除时，所占据的磁盘空间并没有立即被回收。另外，对于那些声明为可变长度的数据列，时间长了会使得数据表出现很多碎片，减慢查询效率。optimize table 语句可以消除删除和更新操作而造成的磁盘碎片，用于回收闲置的数据库空间，从而减少空间浪费。使用了 optimize table 命令后这些空间将被回收，并且对磁盘上的数据行进行重排。optimize table 只对 MyISAM、BDB 和 InnoDB 表起作用，只能优化表中的 varchar、blob 和 text 类型的字段。

对于写比较频繁的表，要定期进行优化，一周或一个月一次，看实际情况而定。

optimize table 语句的基本语法如下：

```
optimize table tablename1[,tablename2,...];
```

通过 optimize table 语句可以消除删除和更新造成的磁盘碎片，从而减少空间的浪费。optimize table 语句在执行过程中也会给表加上只读锁。

例如，利用 optimize table 语句优化 student 表，语句如下：

```
mysql > optimize table student;
```

12.3.5 优化慢查询

MySQL 5.7 支持将执行比较慢的 SQL 语句记录下来。

(1) 查看相关系统变量，查询系统默认状态。

【例 12-12】 执行下面语句并查看系统设置慢查询的标准结果。long_query_time 是用来定义慢于多少秒的才算“慢查询”，系统默认是 10 秒。

代码和运行结果如下：

```
mysql > show variables like 'long%';
        +--------------+----------+
        | Variable_name | Value    |
        +--------------+----------+
        | long_query_time | 10.000000 |
        +--------------+----------+
        1 row in set, 1 warning (0.02 sec)
```

而执行下面语句，可以查看慢查询的设置参数。

代码和运行结果如下：

```
mysql > show variables like 'slow%';
        +-----------------+---------------------+
        | Variable_name   | Value               |
```

```
+-----------------+----------------------+
| slow_launch_time    | 2                    |
| slow_query_log      | on                   |
| slow_query_log_file | PGIG1MIWMYPOFBS-slow.log |
+-----------------+----------------------+
3 rows in set, 1 warning (0.00 sec)
```

说明：

① long_query_time：表示超过多少秒的查询就写入日志，默认是 10 秒，设置为 0 的话表示记录所有的查询。

② slow_query_log：是否打开日志记录慢查询，on 表示打开，off 表示关闭。

③ slow_query_log_file：慢查询日志文件的保存位置，系统默认在"C:\ProgramData\MySQL\MySQL Server 5.7\Data\WMR3RBO"。

(2) 设置变量，优化慢查询。

【例 12-13】 将查询时间超过 1 秒的查询作为慢查询。

代码和运行结果如下：

```
mysql> set long_query_time=1;
       Query OK, 0 rows affected (0.05 sec)
mysql> show variables like 'long%';
       +---------------+---------+
       | Variable_name   | Value    |
       +---------------+---------+
       | long_query_time | 1.000000 |
       +---------------+---------+
       1 row in set, 1 warning (0.00 sec)
```

只要启动慢查询日志记录，一旦 slow_query_log 变量被设置为 on，MySQL 会立即开始记录。

(3) 检查形成慢查询的原因。

① 了解业务方使用场景，查看查询字段是否没有索引或者没有用到索引（这是查询慢最常见的问题，是程序设计的缺陷）。

② 硬件环境问题。内存不足，网络速度慢，I/O 吞吐量小，形成了瓶颈效应。

③ 有没有创建计算列导致查询不优化；查询语句不好，没有优化，是否返回了不必要的数据；查询出的数据量过大（可以采用多次查询，其他的方法降低数据量）。

④ 利用 explain 查看执行计划，是否与预期一致（从锁定记录较少的表开始查询）。

⑤ 是否形成锁或者死锁（这也是查询慢最常见的问题，是程序设计的缺陷）。

⑥ 利用 sp_lock、sp_who 等参数查看活动的用户，是否存在读写竞争资源。

12.3.6 优化表设计

在 MySQL 数据库中，为了优化查询，使查询能够更加精炼、高效，在用户设计数据表的同时，也应该考虑一些因素：

(1) 在设计数据表时应优先考虑使用特定字段长度，后考虑使用变长字段，如在用户创建数据表时，考虑创建某个字段类型为 varchar 而设置其字段长度为 255，但是在实际应用时，该用户所存储的数据根本达不到该字段所设置的最大长度，命令外如设置用户性别的字

段，往往可以用 M 表示男性，F 表示女性，如果给该字段设置长度为 varchar(50)，则该字段占用了过多列宽，这样不仅浪费资源，也会降低数据表的查询效率。适当调整列宽不仅可以减少磁盘空间，同时也可以使数据在进行处理时产生的 I/O 过程减少。将字段长度设置成其可能应用的最大范围可以充分地优化查询效率。

(2) 改善性能的另一项技术是使用 optimize table 命令处理用户经常操作的表，频繁地操作数据库中的特定表会导致磁盘碎片的增加，这样降低 MySQL 的效率，故可以应用该命令处理经常操作的数据表，以便于优化访问查询效率。

(3) 在考虑改善表性能的同时，要检查用户已经建立的数据表，划分数据的优势在于可以使用户更好地设计数据表，但是过多的表意味着性能降低，故用户应检查这些表，检查这些表是否有可能整合到一个表中，如没有必要整合，在查询过程中，用户可以使用连接，如果连接的列采用相同的数据类型和长度，同样可以达到查询优化的作用。

(4) 需要注意的是，数据库表的类型 InnoDB 或 BDB 表处理行存储与 MyISAM 或 ISAM 表的情况不同。在 InnoDB 或 BDB 类型表中使用定长列，并不能提高其性能。

12.4 查询高速缓存

查询缓存用于存储 select 查询的文本以及服务器发送给客户端的相应结果。如果服务器随后收到一个相同的查询，服务器从查询缓存中重新得到查询结果，而不再需要解析和执行查询。

12.4.1 检验高速缓存是否开启

当查询解析之前先进行比较操作，例如"select * from score;"和"Select * From score;"被认为是不同的两个操作，也就意味着查询操作必须是逐字节相同的操作语句串才能够使用高速缓存；同样的查询字符串有可能认为是不同的，如使用不同的数据库、不同的协议版本或者不同的默认字符集的查询，所以高速缓存将建立不同的查询缓冲。

当一个表被更改，那么使用这个表的所有缓存查询将不再有效，并从缓存区中移出。可能更改表的语句包括 insert、update、delete、truncate、alter table、drop table 和 drop database 等命令。

【例 12-14】 在 MySQL 中，应用 variables 关键字，以通配符形式查看服务器变量。

代码和运行结果如下：

```
mysql> show variables like '%query_cache%';
        +------------------------+--------+
        | Variable_name          | Value  |
        +------------------------+--------+
        | have_query_cache       | YES    |
        | query_cache_limit      | 1048576 |
        | query_cache_min_res_unit | 4096  |
        | query_cache_size       | 0      |
        | query_cache_type       | off    |
        | query_cache_wlock_invalidate | off |
        +------------------------+--------+
        6 rows in set, 1 warning (0.01 sec)
```

说明：

本操作主要检验高速缓存是否开启，运行结果的参数的含义如下：

- have_query_cache：表明服务器在默认安装条件下，是否已经配置查询高速缓存。
- query_cache_limit：指定单个查询能够使用的缓冲区大小，默认为 1MB。
- query_cache_min_res_unit：指定分配缓冲区空间的最小单位，默认为 4KB。
- query_cache_size：高速缓存分配空间，如果该空间为 86 则证明分配给高速缓存空间的大小为 86MB。如果该值为 0 则表明查询高速缓存已经关闭。
- query_cache_type：判断高速缓存开启状态，其变量值范围为 0～2。其中当该值为 0 或 off 时，表明查询高速缓存已经关闭；当该值为 1 或 on 时表明高速缓存已经打开；其值为 2 或 demand 时，表明要根据需要运行有 sql_cache 选项的 select 语句，提供查询高速缓存。
- query_cache_wlock_invalidate：如果查询结果位于查询缓存中，则其他客户端未被锁定，可以对该表进行查询。

12.4.2 使用高速缓存

在 MySQL 中，查询高速缓存的具体语法结构如下：

```
select sql_cache * from tablename ;
```

【例 12-15】 查询 student 表高速缓存运行中的反应结果。

代码和运行结果如下：

```
#开启高速缓存
mysql> set @@global.query_cache_size = 1;
       Query OK, 0 rows affected, 1 warning (0.01 sec)
#使用查询高速缓存运行结果
mysql> select sql_cache * from teacher;
       +---------+------+--------+------+----------+
       | teacherno |tname  | major    | prof   | department |
       +---------+------+--------+------+----------+
       | t05001   | 苏超然 | 软件工程 | 教授   | 计算机学院 |
       | t05002   | 常杉   | 会计学   | 助教   | 管理学院   |
       | t05003   | 孙释安 | 网络安全 | 教授   | 计算机学院 |
       | t05011   | 卢敖治 | 软件工程 | 副教授 | 计算机学院 |
       | t05017   | 茅佳峰 | 软件测试 | 讲师   | 计算机学院 |
       | t06011   | 夏南望 | 机械制造 | 教授   | 机械学院   |
       | t06023   | 葛庭宇 | 铸造工艺 | 副教授 | 材料学院   |
       | t06027   | 陶期年 | 纳米技术 | 教授   | 材料学院   |
       | t07019   | 韩既乐 | 经济管理 | 副教授 | 管理学院   |
       +---------+------+--------+------+----------+
       9 rows in set (0.00 sec)
#未使用查询高速缓存运行结果
mysql> select sql_no_cache * from teacher;
       +---------+------+--------+------+----------+
       | teacherno | tname  | major    | prof   | department |
       +---------+------+--------+------+----------+
```

```
| t05001    | 苏超然 | 软件工程 | 教授   | 计算机学院 |
| t05002    | 常杉   | 会计学   | 助教   | 管理学院   |
| t05003    | 孙释安 | 网络安全 | 教授   | 计算机学院 |
| t05011    | 卢敖治 | 软件工程 | 副教授 | 计算机学院 |
| t05017    | 茅佳峰 | 软件测试 | 讲师   | 计算机学院 |
| t06011    | 夏南望 | 机械制造 | 教授   | 机械学院   |
| t06023    | 葛庭宇 | 铸造工艺 | 副教授 | 材料学院   |
| t06027    | 陶期年 | 纳米技术 | 教授   | 材料学院   |
| t07019    | 韩既乐 | 经济管理 | 副教授 | 管理学院   |
+---------+------+--------+------+----------+
9 rows in set (0.00 sec)
```

说明：

如果经常运行查询高速缓存，将会提高 MySQL 数据库的性能。一旦表有变化，使用这个表的查询高速缓存将会失效，且将从高速缓存中删除。这样放置查询从旧表中返回无效数据。另外不使用高速缓存查找可以应用 sql_no_cache 关键字。

12.4.3 优化性能的其他方面

(1) limit 1 可以增加性能。如果知道查询的结果只有一行时，加上 limit 1 可以增加性能，MySQL 数据库引擎会在找到一条数据后停止搜索，而不是继续往后查找下一条符合记录的数据，从而提高查询的效率。

```
mysql > select sname,birthdate from student where sname = '常杉' limit 1;
```

(2) 尽量避免使用“select * from table”，查询时应明确要查询哪些字段。哪些字段是无关的，从数据库里读出的数据越多，越会增加服务器开销，降低查询的效率。

(3) 不要滥用 MySQL 的类型自动转换功能。应该注意避免在查询中让 MySQL 进行自动类型转换，因为转换过程也会使索引变得不起作用。

例如，“select studentno ,daily from score where daily >= '60';”数字 60 就不能写成字符'60'，虽然可以输出正确的结果，但会加重 MySQL 的类型转换任务，使之性能下降。

(4) 尽量避免在 where 子句中对字段进行 null 值判断。null 对于大多数数据库都需要特殊处理，MySQL 也不例外。不要以为 null 不需要空间，其实需要额外的空间，并且，在进行比较的时候，程序会更复杂。当然，这里并不是说就不能使用 null 了，现实情况是很复杂的，依然会有一些情况需要使用 null 值。

(5) 尽量避免在 where 子句中使用“!=”或“<>”操作符。MySQL 只有在使用<，<=，=，>，>=，between 和 like 的时候才能使用索引。尽量避免 where 子句对字段进行函数操作。

```
mysql > select sname from student where year(birthdate) = '1999';
```

可以改为如下形式：

```
mysql > select sname from student where birthdate > = '1999 - 1 - 1' and birthdate < =  '1999 - 12 - 31';
```

(6) 尽量避免 where 子句对字段进行表达式操作。例如：

```
mysql > select studentno from score where score/2 = 40;
```

(7) 尽量避免使用 in 或 not in 操作。对于连续的数值,能用 between 就不要用 in。

```
mysql > select sname from score where score between 60 and 70;
```

12.5 小　　结

本章介绍了数据库优化的含义和查看数据库性能参数的方法,以及优化查询的方法、优化数据库结构的方法和优化 MySQL 服务器的方法。优化数据库结构部分主要介绍了如何对表进行优化。优化 MySQL 服务器涉及很多 MySQL 配置文件和配置文件中的参数。学习本章后,主要掌握如下内容:

- 了解使用索引优化查询的方法。
- 了解如何在 MySQL 中分析查询效率。
- 掌握在 MySQL 中应用高速缓存提高查询性能的方法。
- 掌握如何在多表查询中提高查询性能。
- 掌握在 MySQL 中使用临时表提高优化查询效率的方法。

习　题　12

1. 选择题

(1) 使用 explain 语句可对________语句的执行效果进行分析,通过分析提出优化运行速度的方法。

A. select　　B. insert　　C. delete　　D. create

(2) 多列索引在表的多个字段上创建一个索引。只有查询条件中使用了这些字段中的________时,索引才会被正常使用。

A. 最后 1 个字段　　B. 第 2 个字段　　C. 第 1 个字段　　D. 所有字段

(3) 使用 analyze table 分析表的过程中,数据库系统会对表加一个________。在分析期间,只能读取表中的记录,不能更新和插入记录。

A. 排他锁　　B. 只读锁　　C. 读写锁　　D. 意向锁

(4) 若有某些查询经常涉及多表连接,可以视情况将这些字段建立一个________,来进行查询和统计,提高查询效率。

A. 查询表　　B. 排序表　　C. 中间表　　D. 子查询

(5) 为了解决插入记录时,________过程会降低插入记录速度的情况,在插入记录之前可以先禁用索引,等到记录都插入完毕后再开启索引。

A. 索引　　B. 排序　　C. 查询　　D. 插入

2. 思考题

(1) 如何使用查询缓存区?

(2) 为什么查询语句中的索引有时会没有发挥作用?

(3) 什么是慢查询? 形成慢查询的原因有哪些?

(4) 如何从优化查询的角度进行表字段的设计?

3. 上机练习题(本题利用 teaching 数据库中的表进行操作)

(1) 使用 explain 语句来分析一个查询语句。

(2) 分析查询语句,对比不使用索引和使用索引的情况。

(3) 利用 explain 语句执行查询命令,应用 like 关键字,且匹配字符串中含有的百分号"%"。

(4) 执行 analyze table 语句分析 course 表。

(5) 执行 check table 语句检查表 course。

第 13 章　MySQL 日志文件管理

MySQL 日志是记录 MySQL 数据库的日常操作和错误信息的文件。当数据遭到意外发生丢失时，可以通过日志文件来查询出错原因，并且可以通过日志文件进行数据恢复。因此首先要了解日志的作用，并且掌握各种日志的使用方法和使用二进制日志还原数据的方法。

13.1　MySQL 日志文件简介

日志是 MySQL 数据库不可或缺的重要组成部分。通过分析这些日志文件，可以了解 MySQL 数据库的运行情况、日常操作、错误信息和哪些地方需要进行优化。

1. 日志的特点

MySQL 日志用来记录 MySQL 数据库的运行情况、用户操作和错误信息等。例如，当一个用户登录到 MySQL 服务器时，日志文件中就会记录该用户的登录时间和执行的操作等。或当 MySQL 服务器在某个时间出现异常时，异常信息也会被记录到日志文件中。日志文件可以为 MySQL 管理和优化提供必要的信息。

如果 MySQL 数据库系统意外停止服务，可以通过错误日志查看出现错误的原因。并且可以通过二进制日志文件来查看用户执行了哪些操作，对数据库文件做了哪些修改等。然后根据二进制日志文件的记录来修复数据库。

需要了解的是，启动日志功能会降低 MySQL 数据库的性能。例如，在查询非常频繁的 MySQL 数据库系统中，如果开启了通用查询日志和慢查询日志，MySQL 数据库会花费很多时间记录日志。同时，日志会占用大量的磁盘空间。对于用户量非常大、操作非常频繁的数据库，日志文件需要的存储空间甚至比数据库文件需要的存储空间还要大。

2. 日志文件分类

默认情况下，所有日志创建于 mysqld 数据目录中。通过刷新日志，可以强制 mysqld 来关闭和重新打开日志文件，也可以切换到一个新的日志。当执行一个 flush logs 语句或执行 mysqladmin flush-logs 或 mysqladmin refresh 时，出现日志刷新。如果正使用 MySQL 复制功能，从复制服务器将维护更多日志文件，被称为接替日志。

MySQL 的日志包括二进制日志（ binary log）、错误日志（error log）、通用查询日志（common_query log）和慢查询日志（slow-query log）4 类。除二进制日志外，其他日志都是文本文件。默认情况下，只启动了错误日志的功能。其他 3 类日志都需要数据库管理员进行设置。日志文件通常存储在 MySQL 数据库的数据目录下。4 类日志文件的具体功能如下：

- 二进制日志：以二进制文件的形式记录了数据库中所有更改数据的语句，还可以运用于复制操作。
- 错误日志：记录 MySQL 服务的启动、运行和停止 mysqld 时出现的问题。
- 通用查询日志：记录用户登录和记录查询的信息。
- 慢查询日志：记录所有执行时间超过 long_query_time 秒的查询或不使用索引的查询。

13.2 错误日志

错误日志记载着 MySQL 数据库系统的诊断和出错信息。错误日志文件包含了当 mysqld 启动和停止时，以及服务器运行过程中发生任何严重错误时的相关信息。MySQL 会将启动和停止数据库信息以及一些错误信息记录到错误日志文件中。

13.2.1 启用和设置错误日志

在 MySQL 数据库中，错误日志功能默认是开启的。而且，错误日志无法被禁止。默认情况下，错误日志存储在 MySQL 数据库的数据文件夹下。错误日志文件通常的名称为 hostname.err。其中，hostname 表示 MySQL 服务器的主机名。错误日志的存储位置可以通过 log-error 选项来设置。将 log-error 选项加入到 my.ini 文件的[mysqld]组中，在 Windows 操作系统中形式如下：

```
#my.ini 文件
[mysqld]
log-error=[path/[filename]]
```

说明：

(1) path 为日志文件所在的目录路径。

(2) filename 为日志文件名，修改配置项后，需要重启 MySQL 服务才能生效。

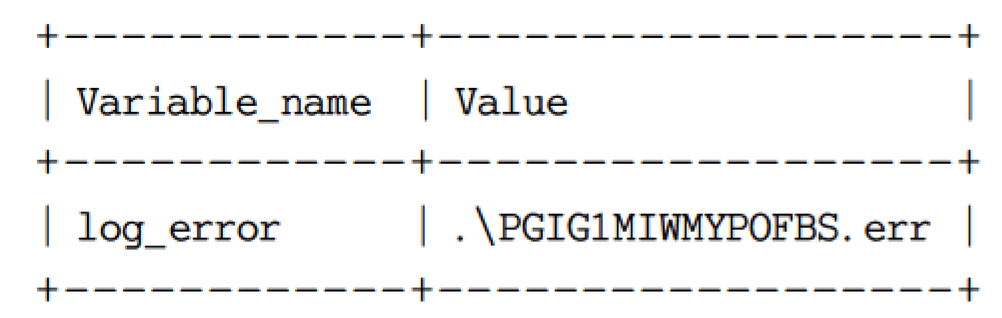
13.2.2 查看错误日志

错误日志中记录着开启和关闭 MySQL 服务的时间，以及服务运行过程中出现哪些异常等信息。如果 MySQL 服务出现故障，可以到错误日志中查找原因。错误日志是以文本文件的形式存储的，可以直接使用普通文本工具就可以查看。Windows 操作系统可以使用文本文件查看器查看。

【例 13-1】 使用记事本查看 MySQL 错误日志。

代码和运行结果如下：

```
#通过 show variables 语句查询错误日志的存储路径和文件名：
mysql> show variables like 'log_error';
```



Variable_name	Value
log_error	.\PGIG1MIWMYPOFBS.err

```
    1 row in set, 1 warning (0.26 sec)
#可以看到错误的文件是\PGIG1MIWMYPOFBS.err,位于 MySQL 默认的数据目录下,使用记事本打开该文件,可以看到 MySQL 的错误日志:
2017-05-22T08:32:31.404282Z 0 [Note] Plugin 'FEDERATED' is disabled.
2017-05-22T08:32:31.404282Z 0 [Warning] Failed to set up SSL because of the following SSL library error: SSL context is not usable without certificate and private key
2017-05-22T08:32:31.404282Z 0 [Note] Server hostname (bind-address): '*'; port: 3306
2017-05-22T08:32:31.404282Z 0 [Note] IPv6 is available.
2017-05-22T08:32:31.404282Z 0 [Note] - '::' resolves to '::';
2017-05-22T08:32:31.404282Z 0 [Note] Server socket created on IP: '::'.
…
2017-05-22T10:05:58.806867Z 6 [Note] Event Scheduler: Dropping mysqltest.direct1
2017-05-22T10:05:58.807867Z 6 [ERROR] Event Scheduler: [root@localhost][mysqltest.direct1] Table 'test1' already exists
```

以上是错误日志文件的一部分,这里面记载了系统的一些错误。

13.2.3 删除错误日志

数据库管理员可以删除很长时间之前的错误日志,以保证 MySQL 服务器上的硬盘空间。MySQL 数据库中,可以使用 mysqladmin 命令来开启新的错误日志。

mysqladmin 命令的语法如下:

```
C:\> mysqladmin -u root -p flush-logs
```

执行该命令后,数据库系统会自动创建一个新的错误日志。旧的错误日志仍然保留着,只是已经更名为 filename.err-old。

如果在客户端登录 MySQL 数据库,可以执行 flush logs 命令:

```
mysql> flush logs;
    Query OK, 0 rows affected (0.11 sec)
```

13.3 二进制日志

二进制日志主要用于记录数据库的变化情况。通过二进制日志可以查询 MySQL 数据库中进行了哪些改变。二进制日志以一种有效的格式,包含了所有更新了的数据或者已经潜在更新了的数据的语句,如没有匹配任何行的一条 delete 语句。语句以事件的形式保存,描述数据的更改。使用二进制日志的主要目的是最大可能地恢复数据,因为二进制日志包含备份后进行的所有更新。



二进制日志

二进制日志包含关于每个更新数据库语句的执行时间信息。它不包含没有修改任何数据的语句。如果要记录所有语句,需要使用通用查询日志。

13.3.1 启用二进制日志

默认情况下,二进制日志功能是关闭的。通过 my.ini 文件的 log-bin 选项可以开启二进制日志。将 log-bin 选项加入到 my.ini 文件的[mysqld]组中。

在 Windows 操作系统中形式如下：

```
#my.ini 文件
[mysqld]
log-bin [ = path\[filename]]
expire_logs_days = 10
max_binlog_size = 100M
```

说明：

(1) log-bin：定义开启二进制的命令关键词。

(2) path 为二进制日志文件所在的目录路径。

(3) filename 为二进制日志文件名。如文件的全名为 filename.000001、filename.000002 等，以此类推。另外还有一个 filename.index 文件，文件内容为所有日志的清单，可以利用记事本方式打开文件。

(4) expire_logs_days：定义清除过期日志的时间，即二进制日志自动删除的天数。

(5) max_binlog_size：定义单个二进制日志文件的大小限制。超出限制，日志就会发生滚动，即关闭当前文件，重新打开一个新的日志文件。该变量的大小范围是 4KB～1GB，如果事务较大，日志文件可能超出 1GB 大小的限制。

在 my.ini 配置文件中[mysqld]下面，添加下列参数，添加完毕启动 MySQL 服务进程，即可启动二进制日志。

```
log-bin
expire_logs_days = 5
max_binlog_size = 10M
```

【例 13-2】 使用 show variables 语句查询日志设置。

代码和运行结果如下：

```
mysql> show variables like 'log_%';
+----------------------------------+----------------------------------+
| Variable_name                    | Value                            |
+----------------------------------+----------------------------------+
| log_bin                          | ON                               |
| log_bin_basename                 | C:\ProgramData\MySQL\MySQL Server5.7
                                   |    \Data\PGIG1MIWMYPOFBS-bin
| log_bin_index                    | C:\ProgramData\MySQL\MySQL Server5.7
                                   |    \Data\PGIG1MIWMYPOFBS-bin.index
| log_bin_trust_function_creators  | OFF                              |
| log_bin_use_v1_row_events        | OFF                              |
| log_builtin_as_identified_by_password | OFF                         |
| log_error                        | .\PGIG1MIWMYPOFBS.err            |
| log_error_verbosity              | 3                                |
| log_output                       | FILE                             |
| log_queries_not_using_indexes    | OFF                              |
| log_slave_updates                | OFF                              |
| log_slow_admin_statements        | OFF                              |
| log_slow_slave_statements        | OFF                              |
| log_statements_unsafe_for_binlog | ON                               |
| log_syslog                       | ON                               |
```

```
| log_syslog_tag                         |                                |
| log_throttle_queries_not_using_indexes | 0                              |
| log_timestamps                         | UTC                            |
| log_warnings                           | 2                              |
+----------------------------------------+--------------------------------+
19 rows in set, 1 warning (0.00 sec)
```

由例题 13-2 的运行结果可以看出，log_bin 的值为 on，表明二进制日志已经启动。

如果想改变日志文件的路径，可以在 my.ini 配置文件中[mysqld]的下面添加下列参数，添加完毕启动 MySQL 服务进程，即可改变二进制日志文件的路径。

```
[mysqld]
log-bin = "D:/mysql/log/binlog"
```

需要注意的是，在实际软件开发和应用过程中，日志文件最好不要和数据文件存放到一个磁盘上，防止出现磁盘故障而无法恢复数据。

13.3.2 查看二进制日志

使用二进制格式可以存储更多的信息，并且可以使写入二进制日志的效率更高。但是，不能直接打开并查看二进制日志。show binary logs 命令可以查看当前的二进制日志文件的个数及其文件名。

【例 13-3】 使用 show binary logs 将查看二进制日志的文件个数及文件名。

代码和运行结果如下：

```
mysql> show binary logs;
    +------------------------+---------+
    | Log_name               | File_size |
    +------------------------+---------+
    | PGIG1MIWMYPOFBS-bin.000001 |   154   |
    +------------------------+---------+
    1 row in set (0.00 sec)
```

查看二进制日志，也可以使用 mysqlbinlog 命令。

mysqlbinlog 命令的语法形式如下：

```
mysqlbinlog filename.number
```

【例 13-4】 使用 mysqlbinlog 查看二进制日志 PGIG1MIWMYPOFBS-bin.000001。

代码和运行结果如下：

```
C:\Users\Administrator> mysqlbinlog C:\ProgramData\MySQL\MySQL Server5.7 \Data\
PGIG1MIWMYPOFBS-bin.000001
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=1*/;
/*!50003 SET @OLD_COMPLETION_TYPE=@@COMPLETION_TYPE, COMPLETION_TYPE=0*/;
DELIMITER /*!*/;
mysqlbinlog: File'C:\ProgramData\MySQL\MySQL' not found
  (Errcode: 2 - No such file or directory)
SET @@SESSION.GTID_NEXT='AUTOMATIC'/* added by mysqlbinlog */ /*!*/;
DELIMITER ;
# End of log file
```

```
/*!50003 SET COMPLETION_TYPE=@OLD_COMPLETION_TYPE*/;
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=0*/;
```

13.3.3 清理二进制日志

二进制日志会记录大量的信息。如果很长时间不清理二进制日志，将会浪费很多的磁盘空间。删除二进制日志的方法如下：

- 删除所有二进制日志。
- 根据编号来删除二进制日志。
- 根据创建时间来删除二进制日志。

（1）删除所有二进制日志。使用 reset master 语句可以删除所有二进制日志。

命令格式如下：

```
reset master;
```

执行完 reset master 命令，所有二进制日志会被删除，MySQL 会重新创建二进制日志文件，新的二进制日志文件重新从 000001 开始编号。

（2）删除指定的日志文件。使用 purge master logs 语句删除指定的日志文件。删除指定文件的语法有以下两种格式：

```
purge {binary| master }logs to'log_name'
purge {binary| master }logs before'date'
```

说明：

① log_name 格式是指定文件名，执行该命令将删除比此文件名编号小的所有二进制日志文件。

② date 是指定日期，执行该命令将删除指定日期以前的所有二进制日志文件。

【例 13-5】 利用 purge master logs 删除创建时间比 PGIG1MIWMYPOFBS-bin.000003 早的所有日志文件。

分析：为了演示语句操作过程，准备多个日志文件，可以对 MySQL 服务进行多次重新启动。例如这里有 5 个日志文件。删除指定文件，可以查看二进制日志文件数，观察到指定文件已经被删除了。

代码和运行结果如下：

```
mysql> show binary logs;
+-----------------------+----------+
| Log_name              | File_size |
+-----------------------+----------+
| PGIG1MIWMYPOFBS-bin.000001 |      177 |
| PGIG1MIWMYPOFBS-bin.000002 |      177 |
| PGIG1MIWMYPOFBS-bin.000003 |      177 |
| PGIG1MIWMYPOFBS-bin.000004 |      177 |
| PGIG1MIWMYPOFBS-bin.000005 |     1207 |
+-----------------------+----------+
5 rows in set (0.00 sec)
mysql> purge master logs TO 'PGIG1MIWMYPOFBS-bin.000003';
```

```
    Query OK, 0 rows affected (0.03 sec)
mysql> show binary logs;
      +------------------------+----------+
      | Log_name               | File_size |
      +------------------------+----------+
      | PGIG1MIWMYPOFBS-bin.000003 |    177 |
      | PGIG1MIWMYPOFBS-bin.000004 |    177 |
      | PGIG1MIWMYPOFBS-bin.000005 |   1207 |
      +------------------------+----------+
      3 rows in set (0.00 sec)
```

【例 13-6】 使用 purge master logs 删除 2017 年 6 月 12 日前创建的所有日志文件。

代码和运行结果如下：

```
mysql> purge master logs before '20170612';
       Query OK, 0 rows affected (0.00 sec)
```

说明：

语句执行之后，2017 年 6 月 12 日之前创建的日志文件都将被删除，但 2017 年 6 月 12 日的日志会被保留，可根据自己机器中创建日志的时间修改命令参数。

13.3.4 利用二进制日志恢复数据库

二进制日志记录了用户对数据库中数据的改变。如 insert、update、delete、create 等语句都会记录到二进制日志中。一旦数据库遭到破坏，可以使用二进制日志来还原数据库。

如果数据库遭到意外损坏，首先应该使用最近的备份文件来还原数据库。备份之后，数据库可能进行了一些更新。这可以使用二进制日志来还原。因为二进制日志中存储了更新数据库的语句，如 update 语句、insert 语句等。

二进制日志还原数据库的命令如下：

```
mysqlbinlog [option] filename | mysql -uuser -ppassword
```

说明：

(1) filename 是日志文件名。

(2) option 是一些可选参数选项，常见的有参数--start-date、--stop-date，用于指定数据库恢复的起始时间点和结束时间点。--start-position、--stop-position 可以指定恢复数据库的开始位置和结束位置。

【例 13-7】 使用 mysqlbinlog 恢复 MySQL 数据库到 2017 年 6 月 12 日 17:00:00 时的状态。

代码和运行结果如下：

```
C:\> mysqlbinlog -stop-date="2017-06-12 17:00:00" C:\Documents and Settings\All Users\
MySQL\MySQL Server 5.7\Data\PGIG1MIWMYPOFBS-bin.000005 -uroot -p
Enter password: ******
```

该命令执行后，会根据指定文件恢复数据库在 2017-06-12 17:00:00 时间以前的所有操作。

13.3.5 暂时停止二进制日志功能

在配置文件中设置了 log-bin 选项以后，MySQL 服务器将会一直开启二进制日志功能。删除该选项后就可以停止二进制日志功能。如果需要再次启动这个功能，又需要重新添加 log-bin 选项。MySQL 中提供了暂时停止二进制日志功能的语句。

如果用户不希望自己执行的某些 SQL 语句记录在二进制日志中，那么需要在执行这些 SQL 语句之前暂停二进制日志功能。

用户可以使用 set 语句来暂停二进制日志功能。该参数的值为 0 时，表示暂停记录二进制日志；如果为 1，则表示恢复记录二进制日志。set 语句的命令格式如下：

```
set sql_log_bin = {0|1};
```

13.4 通用查询日志

通用查询日志用来记录用户的所有操作，包括启动和关闭 MySQL 服务、更新语句、查询语句等。

13.4.1 启动和设置通用查询日志

默认情况下，通用查询日志功能是关闭的。通过 my.ini 文件的 log 选项可以开启通用查询日志。将 log 选项加入到 my.ini 文件的[mysqld]组中，在 Windows 操作系统中的形式如下：

```
#my.ini 文件
[mysqld]
log [ = path\[filename]]
```

说明：

(1) path 为二进制日志文件所在的目录路径。

(2) filename 为通用日志文件名。如果不指定文件名，通用查询日志文件将默认存储在 MySQL 数据目录中的 hostname.log 文件中。hostname 为 MySQL 数据库的主机名。

在不指定参数的情况下，启动通用查询日志的格式如下：

```
[mysqld]
log
```

13.4.2 查看通用查询日志

用户的所有操作都会记录到通用查询日志中。如果希望了解某个用户最近的操作，可以查看通用查询日志。通用查询日志是以文本文件的形式存储的。

Windows 操作系统可以使用文本文件查看器查看。

13.4.3 删除通用查询日志

通用查询日志会记录用户的所有操作。如果数据库的使用非常频繁，那么通用查询日

志将会占用非常大的磁盘空间。数据库管理员可以删除很长时间之前的通用查询日志，以保证 MySQL 服务器上的硬盘空间。

MySQL 数据库中，也可以使用 mysqladmin 命令来开启新的通用查询日志。新的通用查询日志会直接覆盖旧的查询日志，不需要再手动删除了。

mysqladmin 命令的语法如下：

```
mysqladmin -u root -p flush-logs
```

在 Windows 中，服务器打开日志文件期间不能重新命名日志文件。首先，必须停止服务器。然后重新命名日志文件。最后，重启服务器来创建新的日志文件。

13.5 慢查询日志

慢查询日志用来记录执行时间超过指定时间的查询语句。通过慢查询日志，可以查找出哪些查询语句的执行效率很低，以便进行优化。

慢查询日志是记录查询时长超过指定时间的日志。慢查询日志主要用来记录执行时间较长的查询语句。通过慢查询日志，可以找出执行时间较长、执行效率较低的语句，然后进行优化。

13.5.1 启用慢查询日志

默认情况下，慢查询日志功能是关闭的。通过 my.cnf 或者 my.ini 文件的 log-slow-queries 选项可以开启慢查询日志。通过 long_query_time 选项来设置时间值，时间以秒为单位。如果查询时间超过了这个时间值，这个查询语句将被记录到慢查询日志。将 log-slow-queries 选项和 long_query_time 选项加入到 my.ini 文件的[mysqld]组中，在 Windows 操作系统中形式如下：

```
#my.ini
[mysqld]
log-slow-queries [=path\ [filename] ]
long_query_time=n
```

13.5.2 操作慢查询日志

执行时间超过指定时间的查询语句会被记录到慢查询日志中。如果用户希望查询哪些查询语句的执行效率低，可以从慢查询日志中获得想要的信息。慢查询日志也是以文本文件的形式存储的。可以使用普通的文本文件查看工具来查看。

【例 13-8】 查看慢查询日志。使用文本编辑器打开数据目录下的 PGIG1MIWMYPOFBS-slow.log 文件，文件部分如下：

```
\Program Files\MySQL\MySQL Server 5.7\bin\mysqld.exe, Version: 5.7.17-log (MySQL Community
Server (GPL)). started with:
TCP Port: 3306, Named Pipe: (null)
Time    Id Command    Argument
# Time: 2017-05-12T09:21:38.878147Z
```

```
# User@Host: root[root] @ localhost [::1] Id:   3
# Query_time: 59.219387 Lock_time: 0.000000 Rows_sent: 0 Rows_examined: 0
use teaching;
set timestamp = 1494580898;
select exam_iterate(100);
# Time: 2017 - 05 - 12T09:23:37.751947Z
# User@Host: root[root] @ localhost [::1] Id:   5
# Query_time: 20.310162 Lock_time: 0.034002 Rows_sent: 0 Rows_examined: 0
set timestamp = 1494581017;
select exam_iterate (0);
# Time: 2017 - 05 - 12T09:33:23.512450Z
# User@Host: root[root] @ localhost [::1] Id:   7
# Query_time: 90.038150 Lock_time: 0.001000 Rows_sent: 0 Rows_examined: 0
set timestamp = 1494581603;
select exam_iterate(0);
# Time: 2017 - 05 - 12T09:41:25.298007Z
# User@Host: root[root] @ localhost [::1] Id:   9
# Query_time: 55.153155 Lock_time: 0.000000 Rows_sent: 0 Rows_examined: 0
set timestamp = 1494582085;
select exam_iterate(100);
# Time: 2017 - 05 - 12T10:00:52.493766Z
# User@Host: root[root] @ localhost [::1] Id:   11
# Query_time: 24.863422 Lock_time: 0.001000 Rows_sent: 0 Rows_examined: 0
set timestamp = 1494583252;
select exam_iterate(100);
# Time: 2017 - 05 - 12T10:03:10.020633Z
# User@Host: root[root] @ localhost [::1] Id:   13
# Query_time: 48.425770 Lock_time: 0.000000 Rows_sent: 0 Rows_examined: 0
set timestamp = 1494583390;
select exam_iterate(100);
C:\Program Files\MySQL\MySQL Server 5.7\bin\mysqld.exe, Version: 5.7.17 - log (MySQL
Community Server)
```

13.5.3 删除慢查询日志

慢查询日志的删除方法与通用查询日志的删除方法是一样的。可以使用 mysqladmin 命令来删除。也可以使用手工方式来删除。

mysqladmin 命令的语法如下：

```
mysqladmin -u root -p flush-logs
```

执行该命令后，命令行会提示输入密码。输入正确的密码后，将执行删除操作。新的慢查询日志会直接覆盖旧的查询日志，不需要再手动删除了。数据库管理员也可以手工删除慢查询日志。删除之后需要重新启动 MySQL 服务。重启之后就会生成新的慢查询日志。如果希望备份旧的慢查询日志文件，可以将旧的日志文件改名。然后重启 MySQL 服务。

13.6 小 结

本章介绍了日志的含义、作用和优缺点，然后介绍了二进制日志、错误日志、通用查询日志和慢查询日志的内容。二进制日志是本章的难点。二进制日志的查询方法与其他日志不

同，需要读者特别注意。而且，二进制日志可以还原数据库。通过本章的学习，读者应该对如下内容进行重点掌握：

- 二进制日志的启用、查看、暂停和清理等基本操作。
- 利用二进制日志恢复数据库的过程。
- 错误日志的启用、查看和删除等基本操作，设置错误日志存取路径的方法。
- 通用查询日志的启用、查看和删除等基本操作。

习　题　13

1. 选择题

(1) MySQL 的日志在默认情况下，只启动了________的功能。

A. 二进制日志　　B. 错误日志　　C. 通用查询日志　　D. 慢查询日志

(2) MySQL 的日志中，除________外，其他日志都是文本文件。

A. 二进制日志　　B. 错误日志　　C. 通用查询日志　　D. 慢查询日志

(3) 如果很长时间不清理二进制日志，将会浪费很多的磁盘空间。删除二进制日志的方法不包括________。

A. 删除所有二进制日志　　B. 删除指定编号的二进制日志

C. 根据创建时间来删除二进制日志　　D. 删除指定时刻的二进制日志

(4) 如果数据库遭到意外损坏，首先应该使用最近的备份文件来还原数据库，可以使用________来还原。

A. 通用查询日志　　B. 错误日志　　C. 二进制日志　　D. 慢查询日志

2. 思考题

(1) MySQL 的日志分几类，各有什么作用？

(2) 慢查询日志有什么特点和作用？

(3) 简述 MySQL 日志的主要作用。

3. 上机练习题(本题利用 teaching 数据库进行操作)

(1) 使用 show variables 语句查询当前日志设置。

(2) 使用 show binary logs 查看二进制日志文件的个数及文件名。

(3) 使用 purge master logs 删除 2017 年 8 月 30 日前创建的所有日志文件。

(4) 使用记事本查看 MySQL 错误日志。

第 14 章　使用 PHP 操作 MySQL 数据

PHP(Hypertext Preprocessor)即为超级文本预处理语言，是一种集服务器端、跨平台、HTML 嵌入式的脚本语言。PHP 的语法集成了 C 语言、Java 语言和 Perl 语言的特点，是一种被广泛应用的开源式的、多用途的 HTML 内嵌式的脚本语言。PHP 易于学习且能高效地运行于服务器端的优势使之成为目前比较流行的动态网页开发技术。

PHP 提供了标准的数据接口，数据库连接也十分方便，兼容性好，扩展性好，可以进行面向对象编程。尤其适合与 MySQL 搭档，实现编写信息管理系统或开发 Web 网站软件。

基于初学者学习的目的，本书采用 Apache 2.4＋PHP 7.1.6＋MySQL 5.7.17 的框架结构介绍利用 PHP 管理 MySQL 数据库的基本技术。

14.1　初识 PHP 语言

PHP 是目前数据库编程和开发动态网页过程中使用得最为广泛的语言之一，成千上万的网站和组织正以各种形式、多种自然语言提供 PHP 语言的资料、最新的应用和研究成果。

14.1.1　PHP 语言的特点

PHP 能运行在包括 Windows、Linux 等大多数操作系统环境下，常与 Web 服务器软件 Apache 和 MySQL 数据库结合应用于软件开发平台上，成为目前软件开发技术的“黄金组合”，具有非常高的性价比。下面介绍 PHP 开发语言的特点。

(1) 运行速度快。PHP 是一种强大的 CGI 脚本语言，语法混合了 C、Java、Perl 和 PHP 式的新语法，执行网页速度比 CGI、Perl 和 ASP 更快，而且内嵌 Zend 加速引擎，性能稳定。

(2) 支持面向对象技术。PHP 能够使用面向对象编程(OOP)的思想来进行高级编程，提高了 PHP 编程能力，优化了 Web 开发构架。能够实现程序逻辑与用户界面分离。

(3) 经济实用性强。PHP 语法结构简单，易于入门，很多功能只需一个函数就可以实现，并且很多机构都相继推出了用于开发 PHP 的 IDE 工具(例如 NetBeans IDE 8.2)。由于 PHP 是一种面向对象的、完全跨平台的新型 Web 开发语言，所以无论从开发者角度考虑还是从经济角度考虑，都是非常实用的。

(4) 功能强大。PHP 在 Web 项目开发过程中具有极其强大的功能，而且实现相对简单，主要表现在如下几点：

- 可操作多种主流与非主流的数据库，例如 MySQL、SQL Server、Oracle、Access、DB2 等，其中，PHP 与 MySQL 是现在绝佳的组合，可以跨平台运行。

- 可与轻量级目录访问协议进行信息交换。
- 可与多种协议进行通信，包括 IMAP、POP3、SMTP、SOAP 和 DNS 等。
- 使用基于 POSIX 和 Perl 的正则表达式库解析复杂字符串。
- 可以实现对 XML 文档进行有效管理及创建、以及调用 Web 服务等操作。

（5）可选择性。PHP 可以采用面向过程和面向对象两种开发模式，开发人员可以从所开发网站的规模和日后维护等多角度考虑，以选择所开发网站应采取的模式。PHP 进行 Web 开发过程中使用最多的是 MySQL 数据库。PHP 7.1.6 版本中不仅提供了早期 MySQL 数据库操作函数，而且提供了 MySQLi 扩展技术对 MySQL 数据库的操纵，这样开发人员可以从稳定性和执行效率等方面考虑操作 MySQL 数据库的方式。

（6）应用范围广。PHP 具有很好的开放性和可扩展性，属于自由软件，其源代码完全公开，任何程序员为 PHP 扩展附加功能都非常容易。在很多网站上都可以下载到最新版本的 PHP。目前，PHP 主要是基于 Web 服务器运行的，支持 PHP 脚本运行的服务器有多种，其中最有代表性的为 Apache 和 IIS，PHP 不受平台束缚，可以在 Windows、UNIX、Linux 等众多版本的操作系统中架设基于 PHP 的 Web 服务器。目前在互联网上有很多知名网站的开发都是通过 PHP 语言来完成的，例如搜狐、网易和百度等。

（7）版本更新速度快。PHP 几乎每年更新一次，比其他同类软件的更新速度要快得多。

（8）PHP 结合数据库应用的优势。PHP 支持多种数据库，而且提供了与诸多数据库连接的相关函数或类库。在实际应用中，PHP 的一个最常见的应用就是与数据库结合。无论是建设网站还是开发信息系统，都少不了数据库的参与。广义的数据库可以理解成关系型数据库管理系统、XML 文件、甚至文本文件等。

除了使用 PHP 内置的连接函数以外，还可以自行编写函数来间接存取数据库。这种机制给程序员带来了很大的灵活性。

14.1.2 PHP 语言的工作原理

PHP 是基于服务器端运行的脚本程序语言，实现数据库和网页之间的数据交互。一个完整的 PHP 开发环境由以下几个部分构成。

（1）操作系统：网站运行服务器所使用的操作系统。PHP 不要求操作系统的特定性，其跨平台的特性允许 PHP 运行在任何操作系统上，常见的服务器操作系统有 Windows 系列和 Linux 系列（包括 Ubuntu、Red Hat、CentOS 等）。

（2）服务器：搭建 PHP 运行环境时所选择的服务器。PHP 支持多种服务器软件，包括 Apache、IIS、Nginx 等。

（3）PHP 包：用于解析 PHP 脚本文件、访问数据库等，是运行 PHP 代码所必需的软件。

（4）数据库系统：实现系统中数据的存储。PHP 支持多种数据库系统，包括 MySQL、SQL Server、Oracle 及 DB2 等。

（5）浏览器：可以浏览网页。由于 PHP 在发送到浏览器的时候已经被解析器编译成其他的代码，所以 PHP 对浏览器没有任何限制。

用户通过浏览器访问 PHP 网站系统的全过程可以从如图 14-1 所示的描述中更加清晰

地理解，具体包括如下内容：

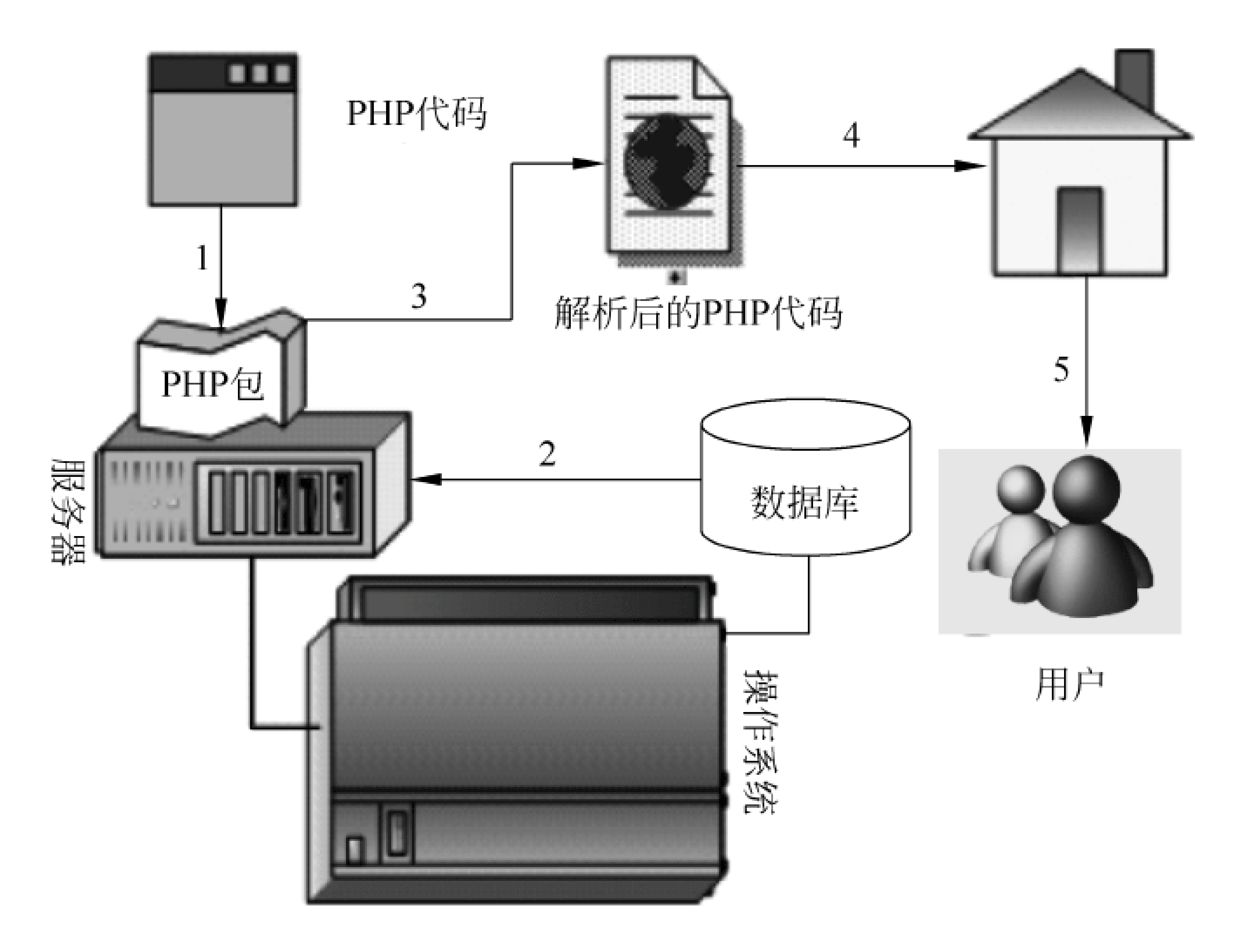


图 14-1　PHP 的工作过程

① PHP 的代码传递给 PHP 包，请求 PHP 包进行解析并编译；

② 在操作系统的支持下，服务器根据 PHP 代码的请求读取数据库；

③ 服务器与 PHP 包共同根据数据库中的数据或其他运行变量，将 PHP 代码解析成普通的 HTML 代码；

④ 解析后的代码发送给浏览器，浏览器对代码进行分析获取可视化内容；

⑤ 用户通过访问浏览器浏览网站内容。

14.2　搭建 PHP+MySQL 的集成开发环境

在运行 PHP 之前，首先需要配置集成开发环境（简称 IDE），集成开发环境是一种为项目开发提供集成环境的应用程序，程序中包含了代码编写、分析调试等工具，方便用户使用。

14.2.1　配置集成开发环境

在编写代码时，IDE 能够进行语法高亮、错误检查、智能补全等辅助操作，可以显著提高工作效率。

在开发 PHP 项目时，常见的 IDE 有 PHPStorm、NetBeans、ZendStudio 等，其中 NetBeans 是一款开源免费的 IDE，功能强大且支持跨平台，推荐使用。NetBeans 支持 Java、PHP、C++等编程语言，本书选用 netbeans-8.2-windows.exe 版本，在 NetBeans 的官方网站（https://netbeans.org）可以下载，安装后的界面如图 14-2 所示。

14.2.2　安装和配置 Apache 软件

Apache HTTP Server（简称 Apache）是 Apache 软件基金会发布的一款 Web 服务器软件，由于其开源、跨平台和安全性的特点被广泛使用。从免费网站 www.apachelounge.



图 14-2　NetBeans 安装完成初始界面

com/download/获取软件，在如图 14-3 所示的网站中找到 httpd-2.4.25-win32-VC14.zip 版本进行下载。

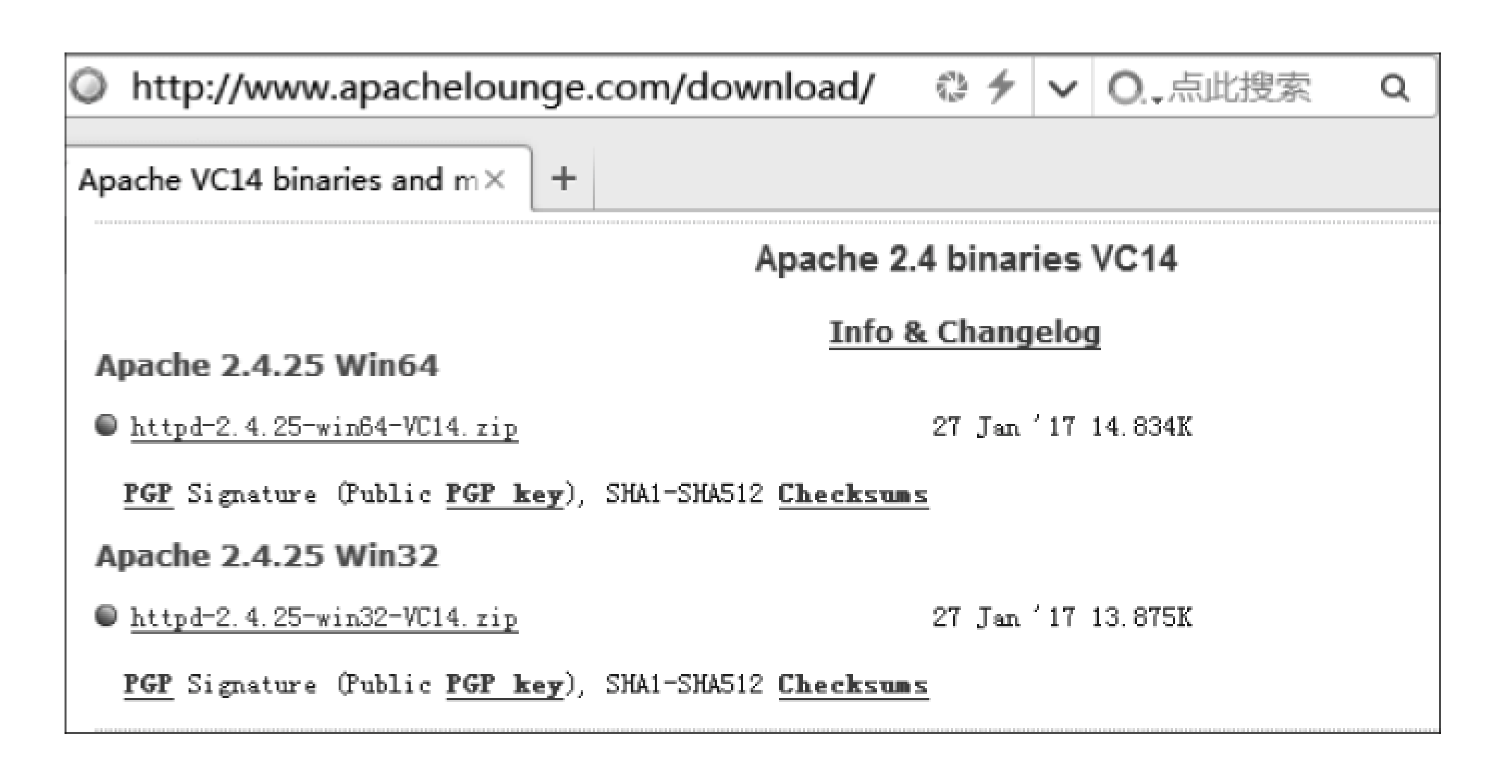


图 14-3　Apache 软件下载

下面以 Apache 2.4 版本为例，讲解 Apache 软件的安装和配置。

1. 安装准备

首先创建 C:\web\apache2.4 作为默认安装目录，然后解压软件 httpd-2.4.25-win32-VC14.zip 压缩包，之后将 apache24 文件夹下的文件剪切到 C:\web\apache2.4 目录下，如图 14-4 所示。

在查看 Apache 目录后，对照表 14-1 了解 Apache 常用目录的功能介绍。重点关注 conf 和 htdocs 两个目录。conf 是服务器的配置目录，包括主配置文件 httpd.conf 和 extra 目录下的若干个辅配置文件。默认情况下辅配置文件是没有开启的。htdocs 是默认站点的网页文档目录，当 Apache 软件服务器启动后，通过浏览器访问本机时，就会查看到 htdocs 目录的网页文档。

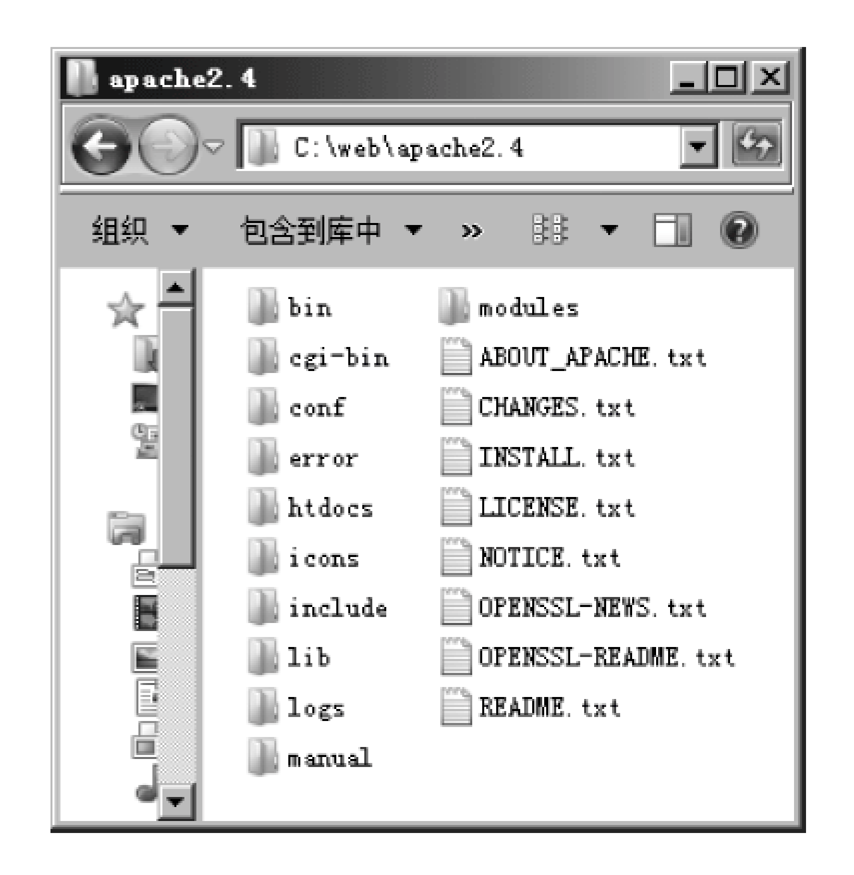


图 14-4 C:\web\apache2. 4 文件夹

表 14-1 Apache 主要目录功能说明

目录名	说　明
bin	Apache 可执行文件目录，如 httpd. exe、ApacheMonitor. exe 等
cig-bin	CGI 网页程序目录
conf	Apache 配置文件目录
htdocs	默认站点的网页文档目录
logs	Apache 日志文件目录，主要包括访问日志 access. log 和错误日志 error. log
manual	Apache 帮助手册目录
modules	Apache 动态加载模块目录

2. 配置 Apache 软件

安装 Apache 软件前，需要先行配置。Apache 软件的配置文件为 conf\httpd. conf，使用 NetBeans 打开该文件，就可以进行下面的配置步骤。

(1) 配置安装目录。如图 14-5 所示，在配置文件中执行文本替换，将 c:/Apache24 全部替换为 c:/web/apache2. 4。

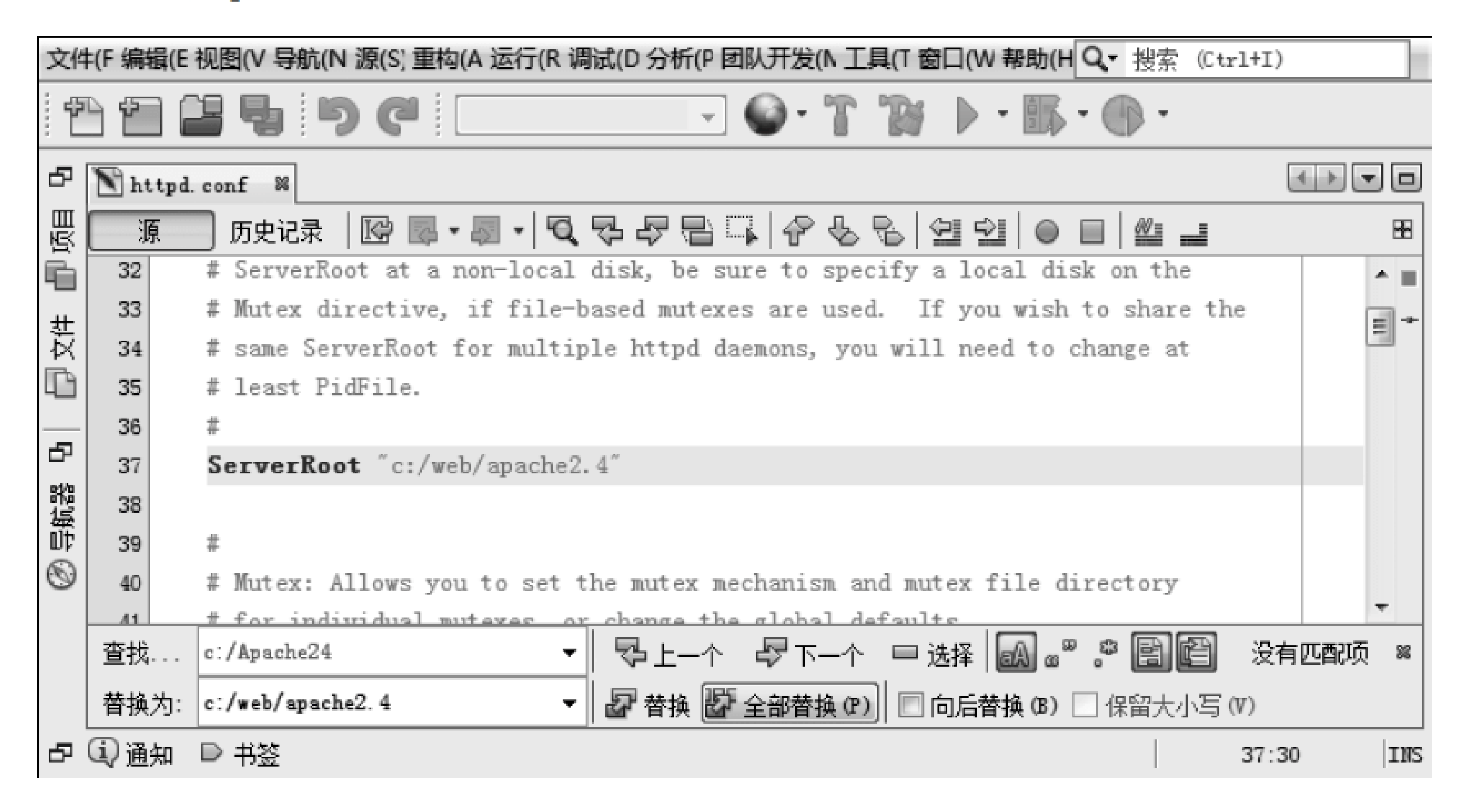


图 14-5 配置安装目录

(2) 配置服务器域名。在图 14-5 所示的界面中搜索 ServerName,找到下面一行配置:

```
#ServerName www.example.com:80
```

其中#表示注释文本,去掉文本注释符号"#"使其生效。

```
ServerName www.example.com:80
```

说明:

经过上述操作后,Apache 已经配置完成。表 14-2 对 Apache 的常用配置进行了解释。

表 14-2　Apache 的常用配置

配　置　项	说　　明
ServerRoot	Apache 服务器的根目录,即安装目录
Listen	服务器监听的端口号,如 80、8080
LoadModule	需要加载的模块
ServerAdmin	服务器管理员的邮箱地址
ServerName	服务器的域名
DocumentRoot	网站根目录
ErrorLog	用于记录错误日志

需要注意的是,一旦修改错误,会造成 Apache 无法安装或无法正常启动,建议在修改前先备份 httpd.conf 配置文件。对于安装目录 ServerRoot 和服务器监听 Listen 的端口号,由于每人的计算机安装的软件不一样,有时会发生端口占用问题,使得 Apache 安装或启动发生异常,此时可以将端口号 80 改成 81,以保证 Apache 的正常使用,此时本地访问方式也会有所不同。

3. 安装 Apache 软件

(1) 启动命令行工具。选择"开始"→"所有程序"→"附件"→"命令提示符"命令,并右击,执行"以管理员身份运行"方式,启动命令行窗口。

(2) 在命令模式下,切换到 Apache 安装目录下的 bin 目录:

```
cd c:\web\apache2.4\bin
```

(3) 输入以下命令代码开始安装(如果需要卸载 Apache,可以使用 httpd.exe -k uninstall 命令进行卸载)。

```
httpd.exe -k install
```

安装 Apache 软件后,就可以作为 Windows 的服务项进行管理了。也可以单击 ApacheMonitor.exe 用于管理 Apache 服务程序在 Windows 系统任务栏右下角状态栏会出现 Apache 的小图标管理工具,通过该图标启动 Apache 服务,当图标由红色变为绿色时,表示启动成功,如图 14-6 所示。

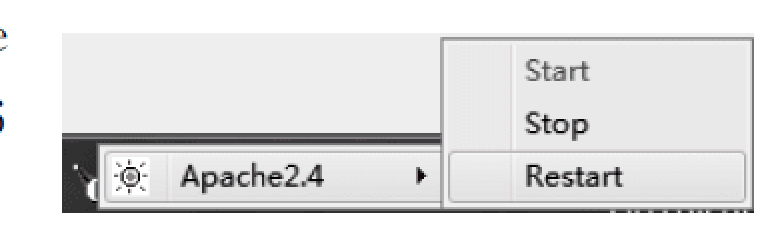


图 14-6　管理 Apache 服务

读者可以将其他网页放到 htdocs 目录下,然后通过"http://localhost/网页文件名"进行访问。

14.2.3 安装和配置 PHP 软件

安装 Apache 软件后，就可以开始 PHP 模块的安装了。PHP 软件有两种安装方式：一种是使用 CGI 应用程序方式，另一种是作为 Apache 服务的模块使用的方式。下面介绍 Apache 服务的模块使用的方式。

1. 获取 PHP 软件

PHP 的官方网站（http://php.net）提供了 PHP 最新版本的下载链接，本书选择 php-7.1.6-Win32-VC14-x86.zip 版本。需要注意的是，PHP 提供了 Thread Safe（线程安全）和 Non Thread Safe（非线程安全）两种选择，在与 Apache 软件匹配时，选择 Thread Safe 版本。

2. 配置 PHP 软件

解压文件。创建 C:\web\php7.1 目录，将下载的 php-7.1.6-Win32-VC14-x86.zip 压缩包解压到该文件夹中，如图 14-7 所示。



图 14-7 PHP 安装目录

其中，PHP 目录结构和主要文件的功能介绍如下：

- ext 是 PHP 扩展文件所在的目录。
- php.exe 是 PHP 的命令行应用程序。
- php5apache2_4.dll 是用于 Apache 的 DLL 模块。
- php.ini-development 是 PHP 预设的配置模板，适用于开发环境。
- php.ini-production 也是配置模板，适合网站上线时使用。

3. 配置 PHP 软件

PHP 提供了开发环境和上线环境的配置模板，模板中有一些选项需要手动配置。

（1）创建配置文件 php.ini。在 PHP 的学习阶段，推荐选择开发环境的配置模板。复制一份 php.ini-development 文件，并命名为 php.ini，该文件将作为 PHP 的配置文件。

（2）配置扩展目录。在 NetBeans 中打开 php.ini，搜索文本 extension_dir 选项找到下面一行配置：

```
;extension_dir = "ext"
```

在 PHP 配置文件中，以分号开头的一行表示注释文本，不会生效。这行配置用于指定 PHP 扩展所在的目录，应将其修改为以下内容：

```
extension_dir = "c:\web\php7.1\ext"
```

(3) 配置 PHP 时区。搜索文本时区选项 date.timezone,找到下面一行配置:

```
;date.timezone =
```

时区可以配置为 UTC(协调世界时)或 PRC(中国时区),配置后如下所示:

```
date.timezone = PRC
```

(4) 在 Apache 中引入 PHP 模块。打开 Apache 配置文件 C:\web\apache2.4\conf\httpd.conf,添加对 Apache 2.4 的 PHP 模块的引入。

```
LoadModule php7_module "c:/web/php7.1/php7apache2_4.dll"
 <FilesMatch "\.php$">
    setHandler application/x-httpd-php
 </FilesMatch>
PHPIniDir "c:/web/php7.1"
```

说明:

- 第 1 行配置表示将 PHP 作为 Apache 的模块来加载。
- 第 2~4 行配置是添加对 PHP 文件的解析,告诉 Apache 服务器将以.php 为扩展名的文件交给 PHP 处理。
- 第 5 行是配置 php.ini 的位置。

配置代码添加后,如图 14-8 所示。

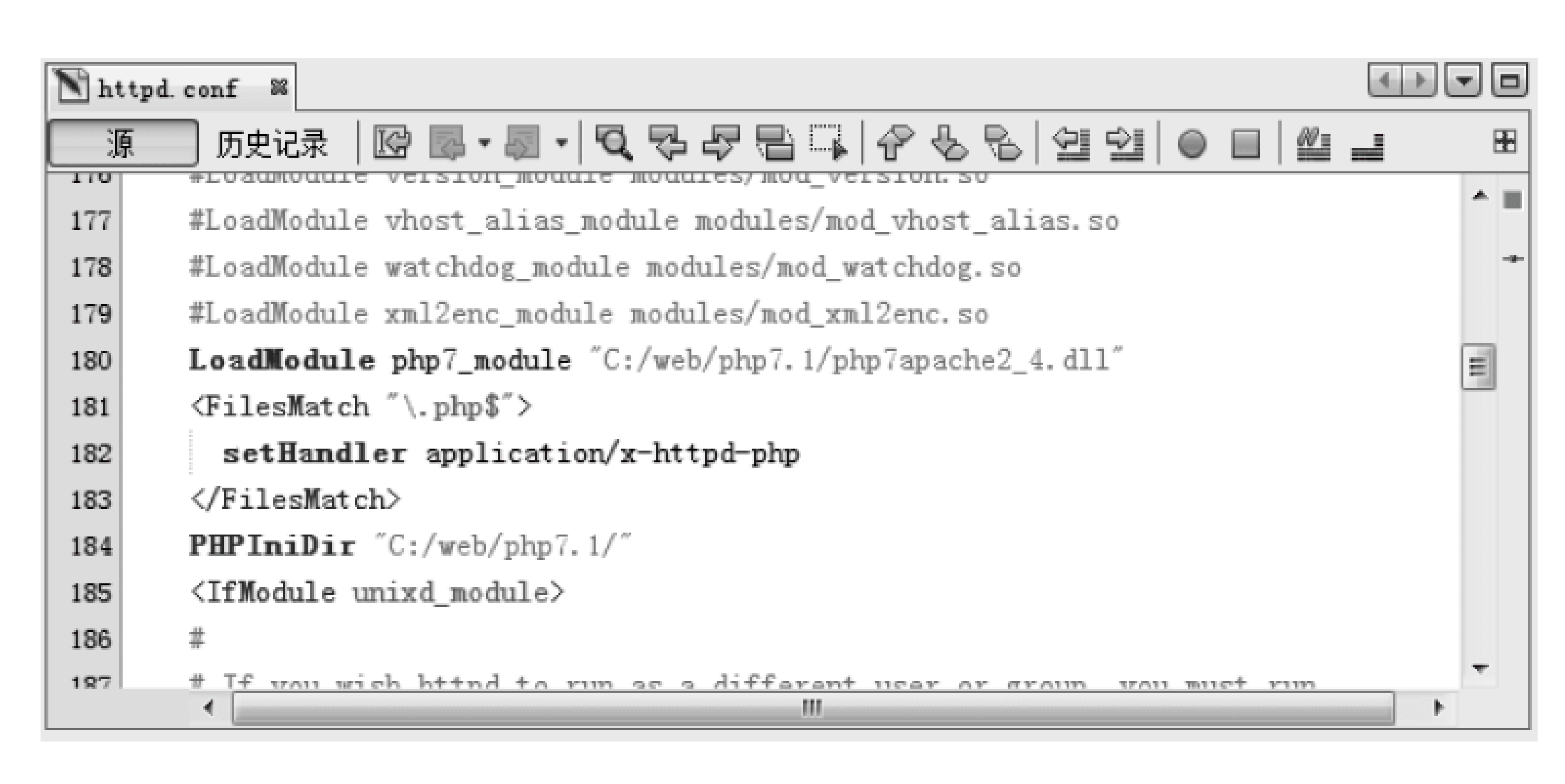


图 14-8 在 Apache 中引入 PHP 模块

(5) 配置 Apache 的索引页。索引页是指访问一个目录时,自动打开哪个文件作为索引页。例如,访问 http://localhost 实际上是访问 http://localhost/index.html,因为 index.html 是默认索引页,所以可以省略文件名。在配置文件 C:\web\apache2.4\conf\httpd.conf 中搜索到 DirectoryIndex,可以找到如下代码:

```
<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>
```

修改为:

```
<IfModule dir_module>
    DirectoryIndex index.html index.php
</IfModule>
```


上述配置表示在访问目录时，首先检测是否存在 index.html，如果有，则显示，否则就继续检查是否存在 index.php。如果一个目录下不存在索引页文件，Apache 会显示该目录下所有的文件和子文件夹(也可以关闭此功能)。

(6) 重新启动 Apache 服务。修改 Apache 配置文件后，需要重新启动 Apache 服务，才能使配置生效。先单击右下角的 Apache 服务图标，选择 Apache2.4 菜单，单击 Restart 命令就可以重启服务。

4. 测试运行 PHP 软件



测试运行 PHP 软件

重启 Apache 服务后，PHP 作为 Apache 的一个模块也一起启动。

(1) 创建测试文件。如果想测试 PHP 是否安装成功，可以在 Apache 的 Web 站点目录 C:\web\apache2.4\htdocs 下使用 NetBeans 创建一个名为 test.php 的文件(也可以直接在该目录下创建此文件)，其内容如下：

```
<?php
    phpinfo();
?>
```

上述代码将 PHP 的配置信息输出到网页中去。代码输入后，如图 14-9 所示。保存文件内容到目录 C:\web\apache2.4\htdocs。

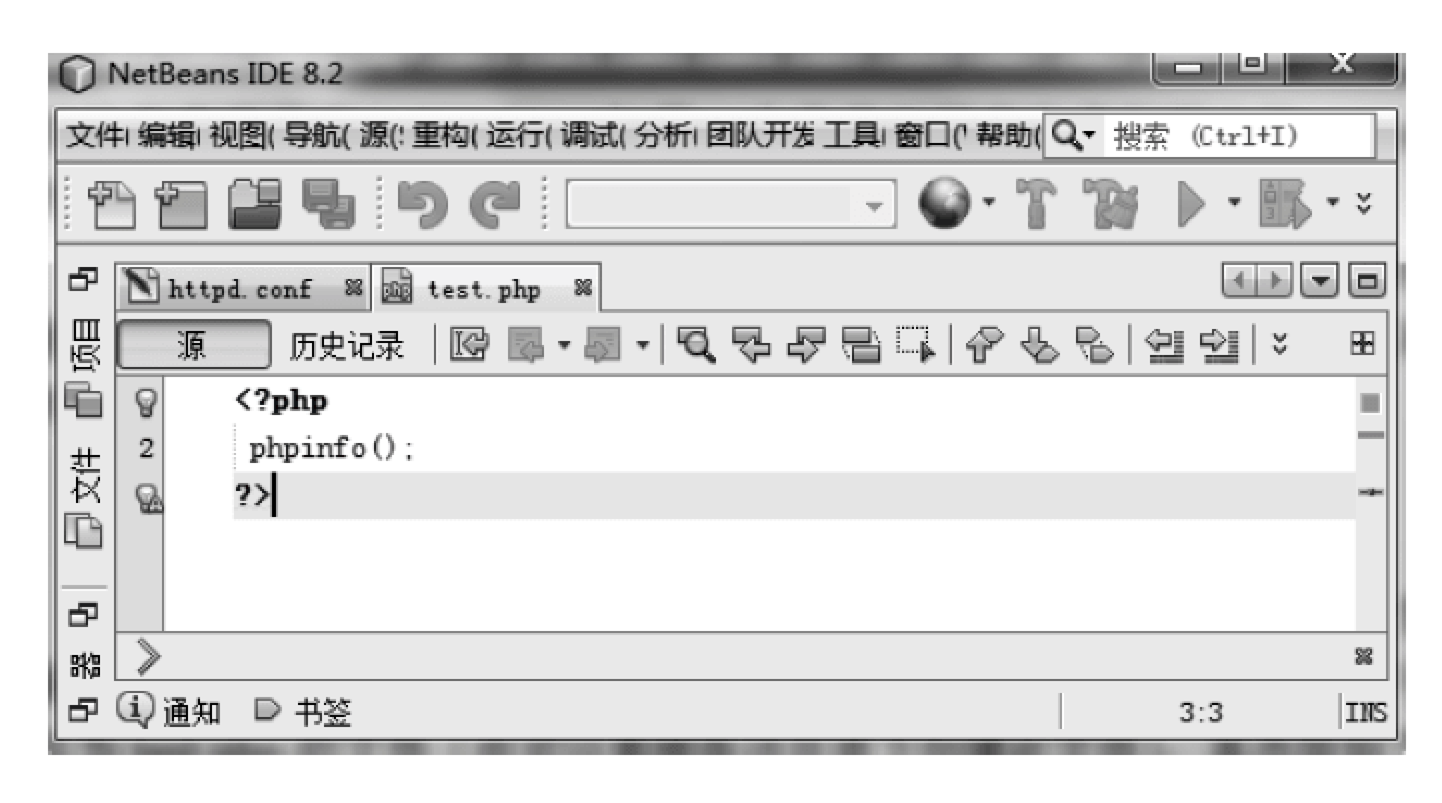


图 14-9　创建测试文件

(2) 测试 PHP 模块是否安装成功。使用浏览器访问地址 http://localhost/test.php，本机端口 80 已经被其他程序占用，故改用 81 端口，浏览器访问地址 http://localhost:81/test.php。如果看到如图 14-10 所示的 PHP 配置信息，说明上述配置成功。否则，需要检查上述配置操作是否有误。

14.2.4　创建 PHP 项目

当 PHP 和 Apache 软件安装配置完毕之后，就可以在 NetBeans 中创建 PHP 项目了。通过 IDE 来管理项目中的代码文件，可以实现编程的可视化。下面介绍如何在 NetBeans

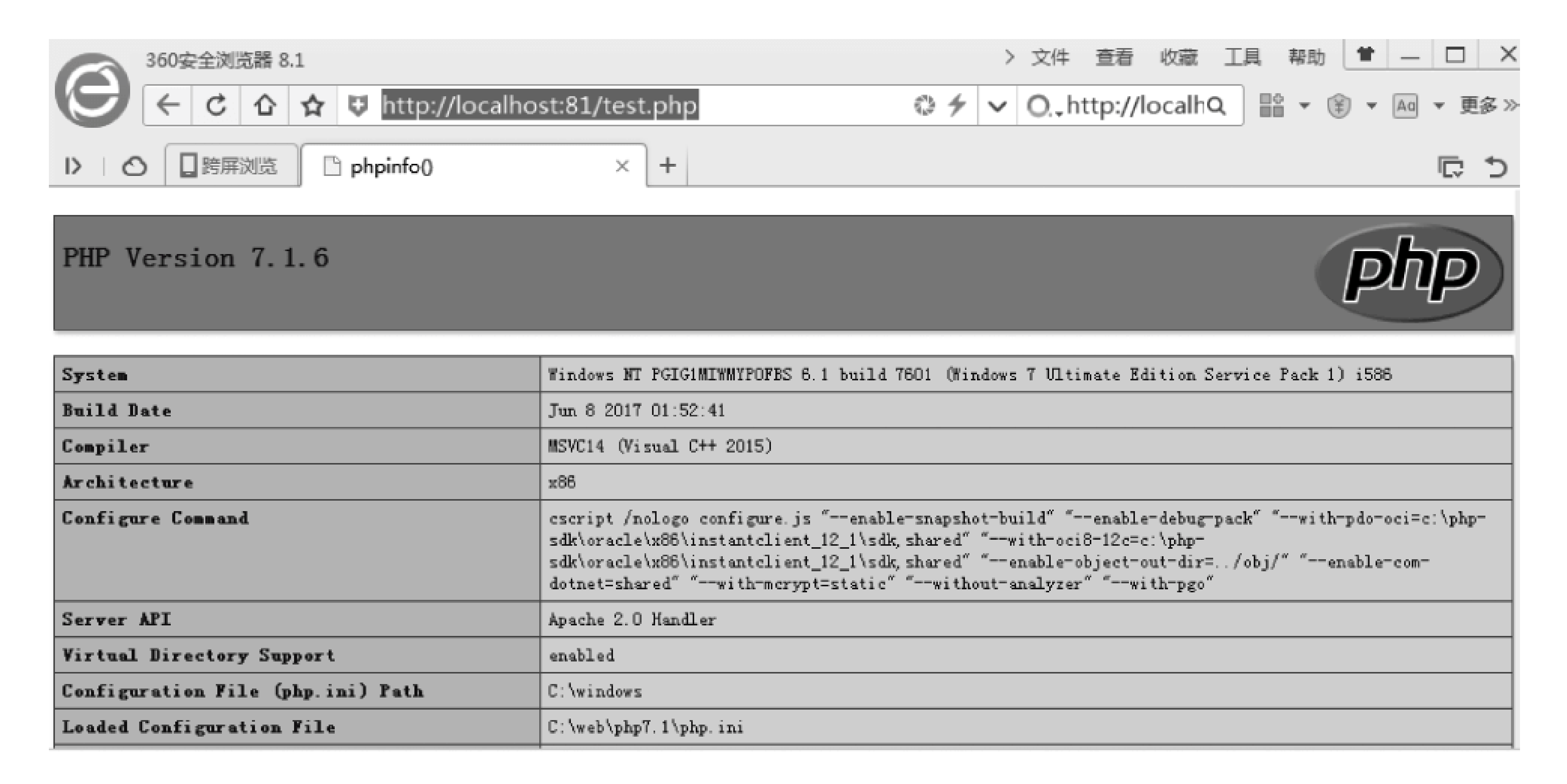


图 14-10　测试 PHP 是否安装成功

中创建 PHP 项目。

(1) 新建项目。在 NetBeans 中执行“文件”→“新建项目”命令，然后选择 PHP 应用程序，如图 14-11 所示。

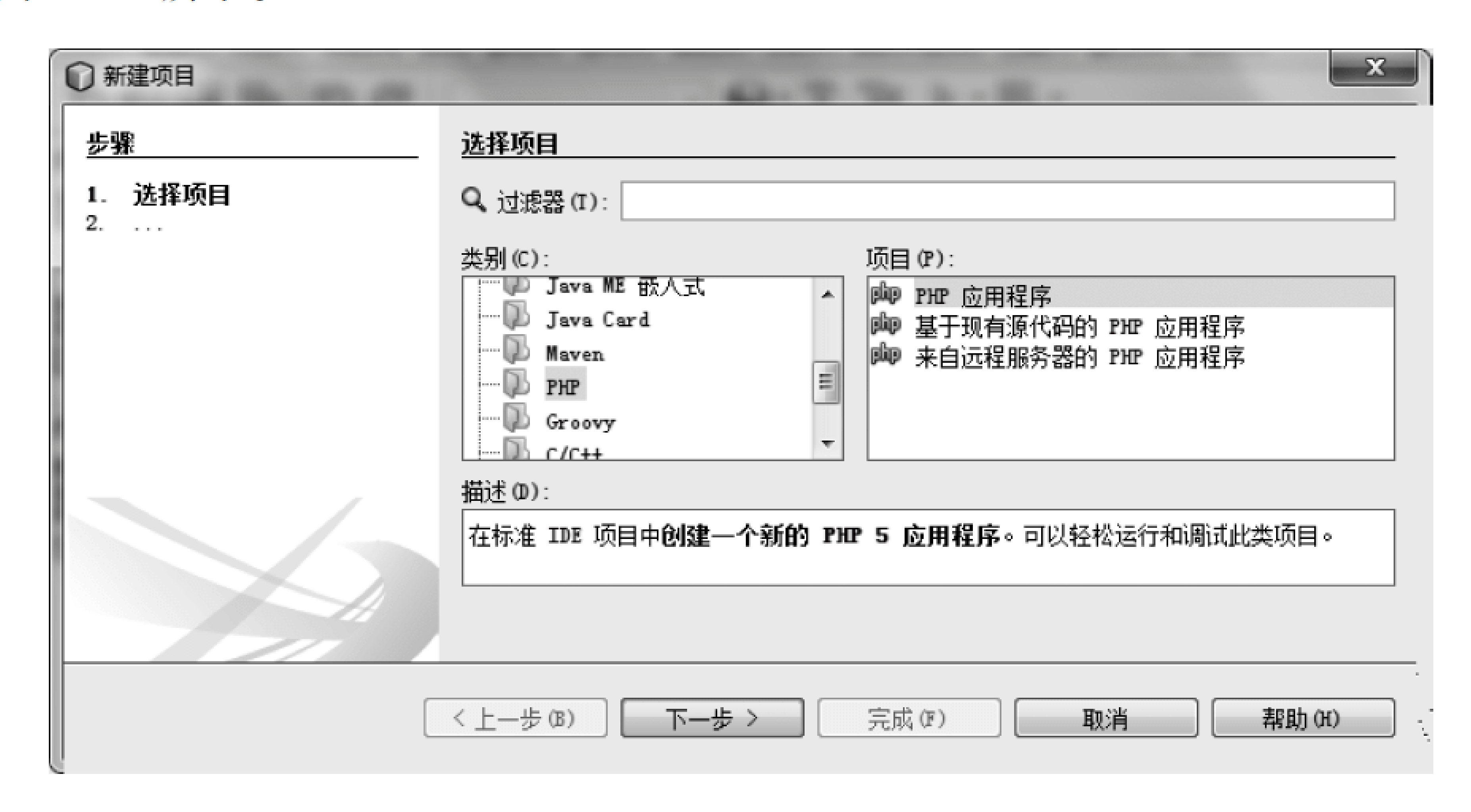


图 14-11　创建 PHP 项目

(2) 配置项目信息。在新建项目的界面单击“下一步”按钮后，开始配置项目的基本信息，如图 14-12 所示，下面设置项目名称和位置参数。

- 项目名称：建议按照项目的特点起名，符合见名知意的原则。
- 源文件夹：选择 Apache 站点目录，即 C:\web\apache2.4\htdocs。
- PHP 版本：用于代码编辑器的语法检查和代码提示，如果考虑项目代码的向下兼容，推荐选择 PHP 7.0 版本。
- 默认编码：常见的编码有 GBK、DTF-8 等。GBK 是国标码，是为了在计算机中处理汉字而设计的编码，只适合中文网站使用，而 DTF-8 支持大多数国家和地区的文字，适合支持国际化的网站应用。

- 将 NetBeans 元数据放入单独的目录：元数据保存项目的基本配置。如果不选中，元数据保存到项目的 nbproject 目录中；如果选中此项，元数据保存到指定目录。



图 14-12　设置 PHP 项目参数

(3) 运行配置。在完成配置项目信息后，单击“下一步”按钮进行运行配置。其中，运行方式选择“本地 Web 站点”，项目 URL 修改为 http://localhost 即可，如图 14-13 所示。



图 14-13　运行配置

(4) 编写代码。在完成配置后，单击“完成”按钮即可创建项目。创建项目后，NetBeans 的界面如图 14-14 所示。可以在 NetBeans 界面中编辑、保存和运行 PHP 文件。

(5) 运行程序。在代码编写完成后，可以通过 NetBeans 自动打开浏览器测试程序，也可以自己在浏览器中输入 URL 地址进行测试。

在 NetBeans 中可以切换浏览器，然后单击绿色三角按钮(快捷键为 F6)运行项目，程序调用浏览器自动访问 index.php。或执行“运行”→“运行文件”命令(快捷键为 Shift+F6)，访问当前编辑的文件。

当然，也可以利用文本文件编写 PHP 文件，还可以通过浏览器执行文件 test1.php。如图 14-15 所示就是该文件的执行结果。

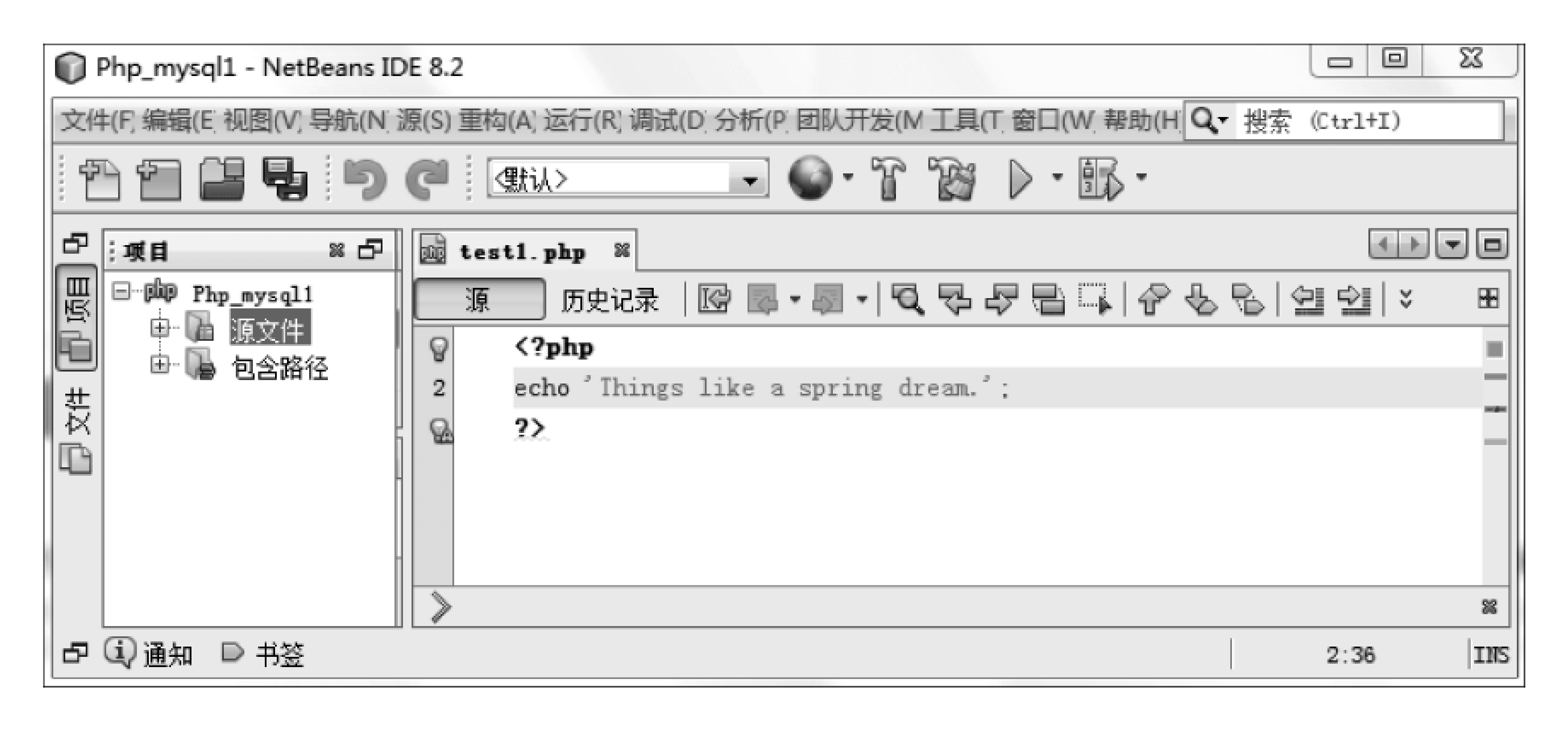


图 14-14　在 NetBeans 界面中编辑 PHP 文件

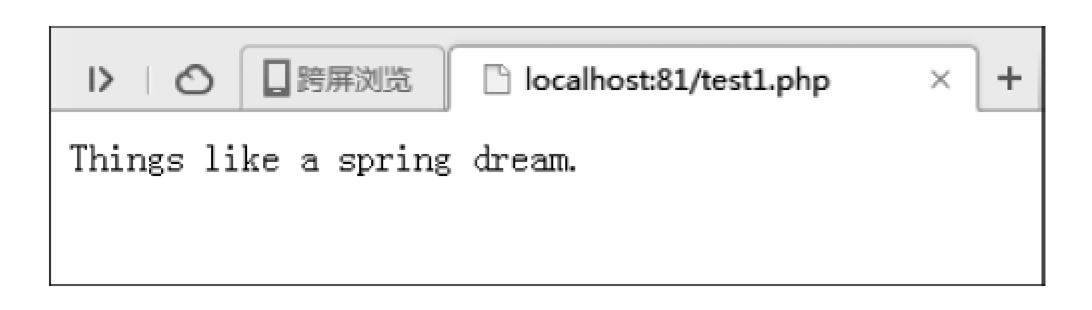


图 14-15　test1.php 的运行结果

14.3　使用 PHP 操作 MySQL 数据库

PHP 7.1 可以通过 mysqlnd 或 mysqli 接口来连接 MySQL 数据库，PHP-MySQL 是 PHP 操作 MySQL 资料库最原始的 Extension（扩展），PHP-MySQLi 的 i 代表 Improvement（改进），提供进阶的功能，就 Extension 而言，本身也增加了安全性。Mysqlnd（MySQL Native Driver）在 PHP 7.1 版本中被作为默认配置选项。从执行 http://localhost:81/test.php 的结果中可以看到 mysqlnd 的参数设置，如图 14-16 所示。



mysqlnd	enabled
Version	mysqlnd 5.0.12-dev - 20150407 - $Id: b396954eeb2d1d9ed7902b8bae237b287f21ad9e $
Compression	supported
core SSL	supported
extended SSL	not supported
Command buffer size	4096
Read buffer size	32768
Read timeout	31536000
Collecting statistics	Yes
Collecting memory statistics	Yes
Tracing	n/a
Loaded plugins	mysqlnd, debug_trace, auth_plugin_mysql_native_password, auth_plugin_mysql_clear_password
API Extensions	no value

图 14-16　mysqlnd 的参数设置

14.3.1 连接 MySQL 服务器

1. 使用 PHP 操作 MySQL 数据库的步骤

PHP 具有强大的数据库支持能力,PHP 操作 MySQL 数据库的步骤如图 14-17 所示。从根本上来说,PHP 是通过预先写好的一些函数来与 MySQL 数据库进行通信的,向数据库发送指令,接收返回数据等都是通过函数来完成的。

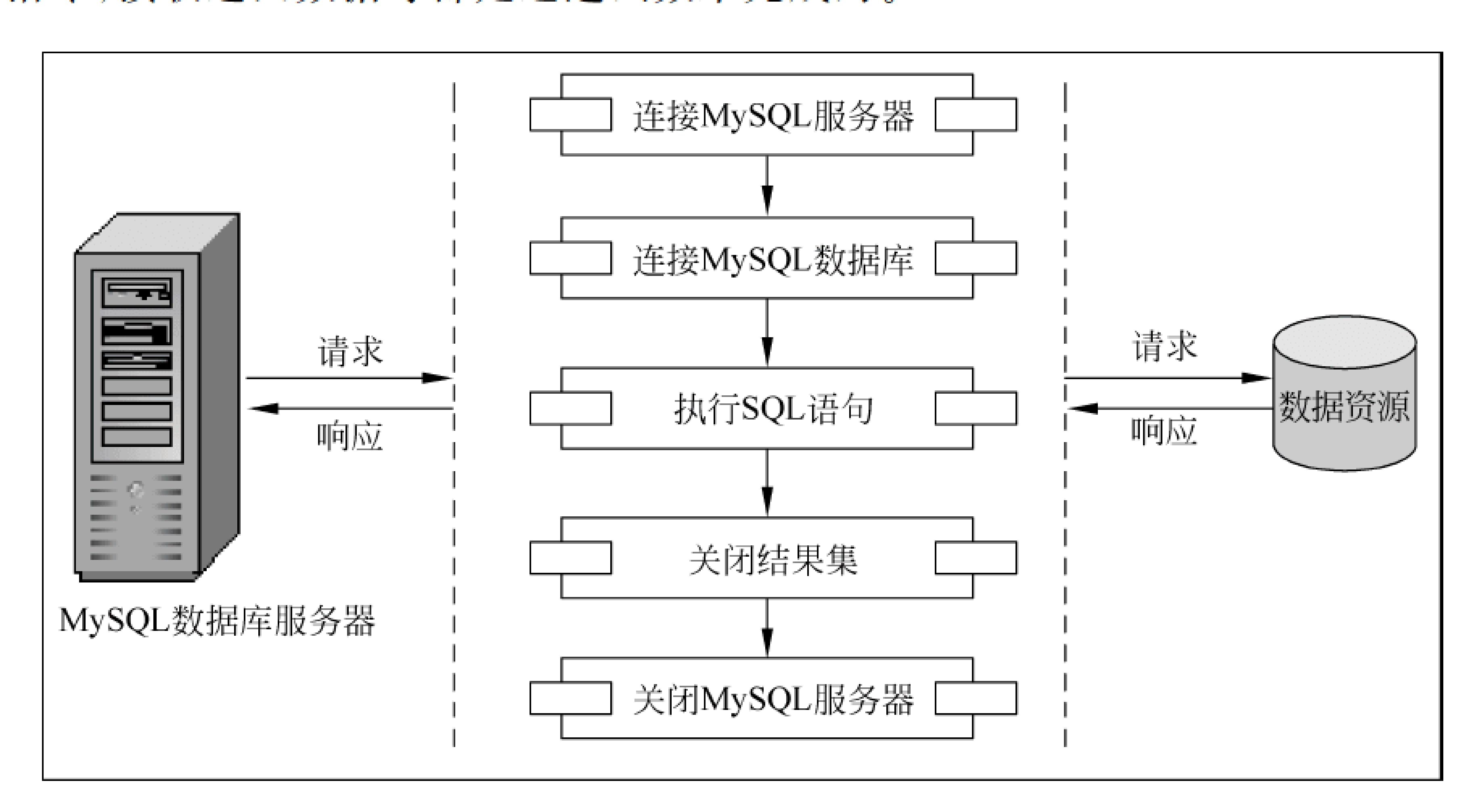


图 14-17 PHP 操作 MySQL 数据库的步骤

PHP 可以通过 MySQL 接口来访问 MySQL 数据库。在 PHP 中加入 MySQL 接口后,才能够顺利地访问 MySQL 数据库。

默认情况下,PHP 不会自动开启对 MySQL 的支持,而是放到扩展函数库中,所以用户需要手动开启 MySQL 函数库。

首先在 NetBeans 中打开 php.ini 文件,查找到下面的选项:

```
;extension = php_mysqli.dll
```

之后,去掉“;”后,保存 php.ini 文件,重新启动 Apache 服务器即可。

2. 利用 mysqli_connect()函数连接 MySQL 服务器

要操作 MySQL 数据库,必须先与 MySQL 服务器建立连接。PHP 中通过 mysqli_connect()函数连接 MySQL 服务器,函数的语法如下:

```
mysqli_connect(hostname,username,password);
```

说明:

(1) mysqli_connect():该函数的返回值用于表示这个数据库连接。如果连接成功,则函数返回一个连接标识,失败则返回 false。

(2) hostname:MySQL 服务器的主机名或 IP,如果省略端口号,默认值为 3306。

(3) username:登录 MySQL 服务器的用户名。

(4) password:MySQL 服务器的用户密码。

【例 14-1】 使用 mysqli_connect()函数连接本地 MySQL 服务器。

PHP 代码如下：

```
<?php
 $ conn = mysqli_connect("localhost", "root", "123456")
     or die("连接数据库服务器失败!".mysql_error());
?>
```

说明：

(1) 为了方便查询因为连接问题而出现的错误，采用 die()函数生成错误处理机制，使用 mysqli_error()函数提取 MySQL 函数的错误文本，如果没有出错，则返回空字符串，如果浏览器显示"Warning: mysqli_connect()..."的字样时，说明是数据库连接的错误，这样就能迅速地发现错误位置，及时改正。

(2) 在 mysqli_connect()函数前面添加符号@，用于限制这个命令的出错信息的显示。如果函数调用出错，将执行 or 后面的语句。die()函数表示向用户输出引号中的内容后，程序终止执行。这样是为了防止数据库连接出错时，用户看到一堆莫名其妙的专业名词，而是提示定制的出错信息。但在调试时不要屏蔽出错信息，避免出错后难以找到问题。

14.3.2 使用 PHP 管理 MySQL 数据库



使用 PHP 管理 MySQL 数据库

1. 使用 mysqli_select_db()函数选择 MySQL 数据库

与 MySQL 服务器建立连接后，可以使用 mysqli_select_db()函数连接 MySQL 服务器中的数据库，函数语法如下：

```
mysqli_select_db(resource link_identifier,databasename)
```

说明：

(1) mysqli_select_db()：连接 MySQL 服务器中的数据库的函数。

(2) resource link_identifier：MySQL 服务器的连接标识。

(3) databasename：选择要连接的 MySQL 数据库名称。

【例 14-2】 连接数据库 teaching，用户名为 root，用户密码为 123456，本地登录。

PHP 代码如下：

```
<?php
 $ conn = mysqli_connect("localhost","root","123456");   //连接 MySQL 数据库服务器
 $ select = mysqli_select_db("teaching", $ conn);        //连接服务器中的 teaching
if( $ select)                                            //判断是否连接成功
    echo "数据库连接成功!";
?>
```

2. 使用 mysqli_query()函数执行 SQL 语句

在 PHP 中，通常使用 mysqli_query()函数来执行对数据库操作的 SQL 语句，包括对数据进行查询、插入、更新和删除等操作。mysqli_query()函数一次只能执行一条 SQL 语句。如果 SQL 语句是 insert、update 和 delete 语句等，语句执行成功，mysqli_query()函数返回 true，否则返回 false。还可以通过 mysqli_affected_rows()函数获取发生变化的记录数。

mysqli_query()函数的语法如下：

```
mysqli_query (resource link_identifier,string query)
```

说明：

(1) 参数 query 是传入的 SQL 语句，包括插入数据(insert)、修改记录(update)、删除记录(delete)、查询记录(select)。

(2) 参数 link_identifier 是 MySQL 服务器的连接标识。

mysqli_affected_rows()函数的语法如下：

```
mysqli_affected_rows(resource link_identifier);
```

【例 14-3】 利用 PHP 语言查询数据表 student 中的数据。

代码和运行结果如下：

```
<?php
$conn1 = mysqli_connect("localhost","root","123456");      //连接 MySQL 数据库服务器
$select = mysqli_select_db($conn1,"teaching");            //连接服务器中的 teaching
if($select){
     header("Content-Type:text/html;charset=gb2312");   //设置字符集
     echo "数据库连接成功!" ;                              //判断是否连接成功
          }
$query = "select * from student";
$result = mysqli_query($conn1, $query) or die("查询失败!".mysqli_error());
echo mysqli_affected_rows($conn1);
?>
```

运行结果：

```
数据库连接成功!11
```

【例 14-4】 向 score 表插入数据。

主要代码如下：

```
$sqlinsert = "insert into score values('19126113307','c05108',80,90)";
$result = mysqli_query($conn1, $sqlinsert)
   or die("插入失败!".mysqli_error());
```

【例 14-5】 删除 score 表数据。

主要代码如下：

```
$sqldelete = "delete from score where studentno = '19126113307' and courseno = 'c05108'";
mysqli_query($conn1, $sqldelete);
```

【例 14-6】 更新 score 表数据。

主要代码如下：

```
$sqldelete = " update score set final = 99 where studentno = '19126113307' and courseno =
'c06108'";
mysqli_query($conn1, $sqldelete);
```

如果需要一次执行多个 SQL 语句，需要使用 mysqli_multi_query()函数。

【例 14-7】 通过 multi_query()函数来执行多条 SQL 语句。

分析：具体做法是，把多条 SQL 命令写在同一个字符串里作为参数传递给 multi_

query()函数,多条 SQL 之间使用分号分隔。如果第一条 SQL 命令在执行时没有出错,这个方法就会返回 true,否则将返回 false。将字符集设置为 GB2312,并向 score 表插入一行数据,然后查询 score 表数据,代码如下:

```
$query = "insert into score values('19126113307','c05108',80,90);";
                                                //向 score 表插入一行数据
$query = "select * from score;";                //设置查询 score 表数据
mysqli_multi_query($conn1, $query);
```

3. 使用 mysqli_close()函数关闭连接

每使用一次 mysqli_connect()或 mysqli_query()函数,都会消耗系统资源。在少量用户访问 Web 网站时问题还不大,但如果用户连接超过一定数量,就会造成系统性能的下降。为了避免这种现象的发生,在完成数据库的操作后,应使用 mysqli_close()函数关闭与 MySQL 服务器的连接,以节省系统资源。mysqli_close()函数的语法如下:

```
mysqli_close($conn);
```

说明:

在 Web 网站的实际项目开发过程中,经常需要在 Web 页面中查询数据信息。查询后使用 mysqli_close()函数关闭数据源。

14.3.3 使用 PHP 处理 MySQL 结果集



使用 PHP 处理 MySQL 结果集

1. 在 PHP 中定义数组

(1) PHP 数组的概念。PHP 中的数组是存储一组数据的集合。数组中的数据称为数组元素,通过"键=>值"形式表示。具体的表示方法如下:

- "键"是数组元素的识别名称,也被称为数组下标。
- "值"是数组元素的内容,"键"和"值"之间使用"=>"连接数组。
- 各个元素之间使用逗号","分隔,最后一个元素后面的逗号可以省略。

PHP 中的数组根据下标的数据类型可分为索引数组(图 14-18(a))和关联数组(图 14-18(b))。索引数组是下标为整型的数组,默认下标从 0 开始,也可以自己指定。关联数组是下标为字符串的数组。

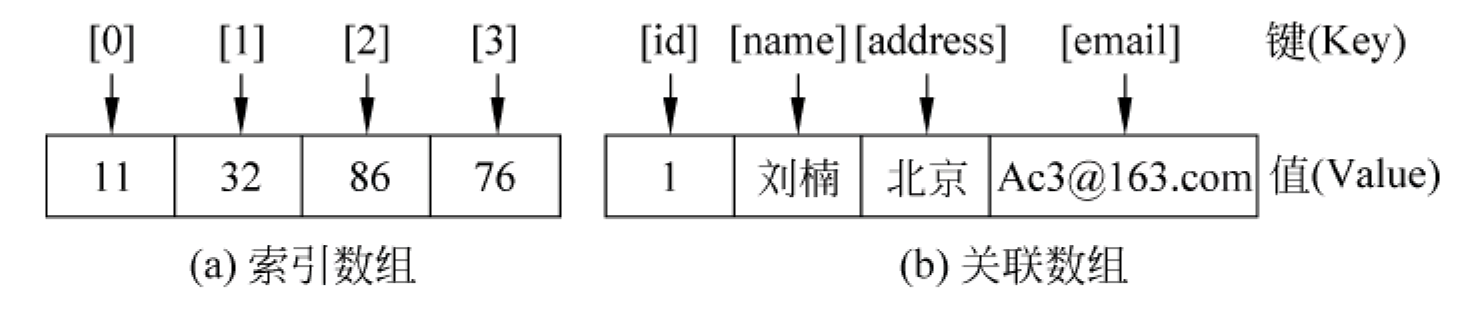


图 14-18 数组存储结构的示意图

(2) 定义 PHP 数组。在使用数组前,首先需要定义数组,在 PHP 中可以使用 array()进行定义。数组中的元素通过"键=>值"的形式表示,各个元素之间使用逗号分隔。

```
//定义关联数组
$card = array('id'=>100, 'name'=>'Tom');     //使用字符串作为键
//定义索引数组
$color = array('red', 'blue');               //省略键时,默认使用 0、1 作为键
```

```
$ fruit = array(2 =>'apple', 5 =>'grape');      //指定下标
//定义空数组、混合型数组
$ empty = array();                              //空数组
//数组元素支持多种数据类型和多维数组
$ mixed = array(0,'str', true, array(1, 2));
$ data = array('name' =>'test', 123);           //此时 123 省略键,默认使用 0 作为键
$ list = array(5 =>'a', 'id' =>'b', 123);       //此时 123 省略键,默认使用 6 作为键
```

从 PHP 5.4 版本起,新增了定义数组的简写语法“[ ]”,使用“[ ]”定义数组的语法与array()语法类似,书写更加方便。

```
$ color = ['red','blue'];                       //相当于: array('red', 'blue')
$ fruit = ['a' =>'apple','b' =>'grape'];
//相当于: array('a' =>'apple', 'b' =>'grape')
$ number = [[1,2],[3,4]];                       //相当于: array(array(1,2), array(3,4))
```

在定义数组时,还需要注意以下几点:

- 数组元素的下标只有整型和字符串两种类型,如果有其他类型,则会进行类型转换。
- 在 PHP 中合法的整数值下标会被自动地转换为整型下标。
- 若数组存在相同的下标时,后面的元素值会覆盖前面的元素值。

(3) 访问 PHP 的数组元素的方法。

① echo()显示函数在前面的内容中已经使用过,用于输出一个或多个字符串。

单引号:定义字符串最简单的方法是用单引号括起来。如果要在字符串中表示单引号,则需要用转义符“\”将单引号转义之后才能输出。与其他语言一样,如果在单引号之前或者字符串结尾处出现一个反斜线“\”,就要使用两个反斜线来表示。

双引号。使用双引号将字符串括起来同样可以定义字符串。如果要在定义的字符串中表示双引号,则同样需要用转义符转义。

② print_r()函数显示关于一个变量的易于理解的信息。如果给出的是 string、integer 或 float,将打印变量值本身。如果给出的是 array,将会按照一定格式显示键和元素。记住,print_r()将把数组的指针移到最后边。使用 reset()可让指针回到开始处。

在开发过程中,若要获取数组中的某个元素,或想要查看数组中的所有元素,可以通过 print_r()或 echo()函数实现。还可以通过 var_dump()函数输出一个变量的详细信息,相关内容可以通过网络查询。例如:

```
//定义数组
$ info = ['id' => 5, 'name' =>'Tom'];
//通过键名访问元素
echo $ info['name'];                            //输出结果: Tom
$ var = 'id';                                   //也可以使用变量的值作为键名
echo $ info[ $ var];                            //输出结果: 5
//通过 print_r()或 var_dump()输出结果
print_r( $ info);                               //输出结果: Array([id] => 5 [name] => Tom )
var_dump( $ info); //输出结果: array(2){ ["id"] => int(5) ["name"] => string(3)"Tom" }
```

(4) 数组赋值。数组赋值的方式和访问数组类似,键名可以省略,省略时自动使用数字索引。

```
$ arr = [];                                     //定义数组(此步骤也可以省略)
```

```
$arr[] = 'PHP';                    //等价于: $arr[0] = 'PHP'
$arr[] = 'Java';                   //等价于: $arr[1] = 'Java'
$arr[5] = 'C语言';                 //等价于: $arr[5] = 'C语言';
$arr['sub'] = 'iOS';               //等价于: $arr['sub'] = 'iOS';
$arr[] = 'HTML';                   //等价于: $arr[6] = 'HTML'
$arr[6] = 'Javaee';                //修改数组,替换已经存在的元素
```

经过上述赋值后,数组的完整结构为:

```
$arr = [0=>'PHP',1=>'Java', 5=>'C语言','sub'=>'iOS','6'=>'Javaee']
```

2. 使用mysqli_fetch_array()函数将结果集返回到数组中

使用mysqli_query()函数执行select语句时,将成功返回查询结果集,返回结果集后,使用mysqli_fetch_array()函数可以获取查询结果集信息,并放入到一个数组中,函数语法如下:

```
array mysqli_fetch_array(result [,int result_type])
```

说明:

(1) 参数result:资源类型的参数,要传入的是由mysqli_query()函数返回的数据指针。

(2) 参数result_type:可选项,设置结果集数组的表述方式,默认值是mysqli_both。其可选值如下:

- mysqli_assoc:表示数组采用关联索引。
- mysqli_num:表示数组采用数字索引。
- mysqli_both:同时包含关联和数字索引的数组。

3. 使用mysqli_fetch_row()函数从结果集中获取一行作为枚举数组

mysqli_fetch_row()函数从结果集中取得一行作为枚举数组。在应用mysqli_fetch_row()函数逐行获取结果集中的记录时,只能使用数字索引来读取数组中的数据,其语法如下:

```
array mysqli_fetch_row (result)
```

说明:

(1) mysqli_fetch_row()函数返回根据所取得的行生成的数组,如果没有更多行则返回false。返回数组的偏移量从0开始,即以$row[0]的形式访问第一个元素(只有一个元素时也是如此)。

(2) resource result:资源类型的参数,要传入的是由mysqli_query()函数返回的数据指针。

4. 使用mysqli_num_rows()函数获取查询结果集中的记录数

使用mysqli_num_rows()函数可以获取由select语句查询到的结果集中行的数目,mysqli_num_rows()函数的语法如下:

```
int mysqli_num_rows(result)
```

说明:

此命令仅对select语句有效。要取得被insert、update或者delete语句所影响到的行

的数目，要使用 mysqli_affected_rows()函数。

5. 使用 mysqli_fetch_assoc()函数从结果集中取得一行作为关联数组

使用 mysqli_fetch_assoc()函数从 select 语句查询到的结果集中取得一行作为关联数组。mysqli_fetch_assoc()函数的语法如下：

```
mysqli_fetch_assoc(result);
```

说明：

(1) 该函数返回的字段名是区分大小写的。

(2) result 是必需的参数，需是 mysqli_use_result()、mysqli_store_result()或 mysqli_query()返回的结果集标识符。

14.3.4 使用 mysqli_free_result()函数释放内存

mysqli_free_result()函数用于释放内存，数据库操作完成后，需要关闭结果集，以释放系统资源，该函数的语法如下：

```
mysqli_free_result(result);
```

说明：

mysqli_free_result()函数将释放所有与结果标识符 result 所关联的内存。该函数仅需要在考虑到返回很大的结果集时会占用多少内存时调用。在脚本结束后所有关联的内存都会被自动释放。

14.3.5 关闭创建的对象

对 MySQL 数据库的访问完成后，必须关闭创建的对象。连接 MySQL 数据库时创建了 $connection 对象，处理 SQL 语句的执行结果时创建了 $result 对象。操作完成后，这些对象都必须使用 close()方法来关闭。

利用 mysqli_close()关闭数据库对象的基本形式如下：

```
mysqli_close(connect);
```

说明：

connect 为连接标识符。

【例 14-8】 查询课程号为 c08171 的成绩信息，并利用输出 echo 命令和 print_r()函数两种方式输出。

PHP 代码和运行结果如下：

```
<?php
$conn1 = mysqli_connect("localhost","root","123456");     //连接 MySQL 数据库服务器
$select = mysqli_select_db( $conn1,"teaching");           //连接服务器中的 teaching
if( $select){
   header("Content - Type:text/html;charset = gb2312");   //设置字符集
   echo "数据库连接成功!" ;                                  //判断是否连接成功
}
$sql  =  "select  *  from score where courseno = 'c08171';";
```

```
if ( $ result = mysqli_query( $ conn1, $ sql))
   {
      while ( $ row = mysqli_fetch_assoc( $ result))
        {
          echo "< br />";
          echo "echo 格式: "."< br />";
          echo "学号 ". $ row['studentno'];
          echo " 课程号 ". $ row['courseno'];
          echo " 平时成绩". $ row['daily'];
          echo " 期末成绩". $ row['final']."< br/>";
          echo "print_r()函数格式: "."< br />";
          print_r( $ row);
        }
      mysqli_free_result( $ result);                    //释放内存
    }
  mysqli_close( $ conn1);                               //关闭连接对象
?>
```

运行结果如下：

```
数据库连接成功!
echo 格式:
学号 18125111109 课程号 c08171 平时成绩 77.0 期末成绩 92.0
print_r()函数格式:
Array ( [studentno] => 18125111109 [courseno] => c08171 [daily] => 77.0 [final] => 92.0 )
echo 格式:
学号 18135222201 课程号 c08171 平时成绩 95.0 期末成绩 82.0
print_r()函数格式:
Array ( [studentno] => 18135222201 [courseno] => c08171 [daily] => 95.0 [final] => 82.0 )
echo 格式:
学号 18137221508 课程号 c08171 平时成绩 88.0 期末成绩 98.0
print_r()函数格式:
Array ( [studentno] => 18137221508 [courseno] => c08171 [daily] => 88.0 [final] => 98.0 )
```

14.4 常见问题与解决方法

在利用 PHP 访问 MySQL 数据库的过程中，除了代码本身、数据格式等因素的错误外，因为硬件环境、软件配置的差异，常常还会碰到很多意想不到的问题。下面对几个常见的问题的处理方法进行介绍。

1. MySQL 服务器无法连接

MySQL 服务器无法连接的错误信息如下所示：

```
Warning:mysqli_connect ( ) [function. mysql - connect]: Unkown MySQL server host 'localhost'
(11001) in E:\wamp\www\test.php on line 2
```

(1) 出现这条错误信息的原因可能有以下几点：

- 代码中的 mysqli_connect 函数中指定的服务器地址有误。
- 数据库服务器不可用。

（2）解决方案如下：

- 检查代码中的服务器地址是否正确。
- 检查数据库服务器是否已经启动并且可用。

2. 用户无权限访问 MySQL 服务器

用户无权限访问 MySQL 服务器的错误信息如下：

```
Warning:mysqli_connect() [function.mysql - connect]: Access denied for user 'root'@ 'localhost
'(using password:NO) in E:\wamp\www\test.php on line 2
```

出现这条错误信息的原因可能是代码中的 mysqli_connect 函数中能够指定的用户名或者密码有误或者在当前服务器上不可用。此类错误的解决方案如下：

- 检查代码中的用户名和密码是否正确。
- 通过 MySQL 命令行测试是否可以使用该用户名和密码登录 MySQL 数据库服务器。

3. 提示 mysqli_connect 等函数未定义

提示 mysqli_connect 等函数未定义的错误信息如下：

```
Fatal error : Call to undefined function mysqli_connect() in E:\wamp\www\test.php on line 2
```

出现这条错误信息的原因可能是在 php.ini 文件中没有配置 MySQL 的扩展库。一般的解决方案是：编辑 php.ini 文件，定位到如下位置，去掉此项前面的分号，保存后重新启动 Apache 服务器。

```
;extension = php_mysql.dll
```

4. SQL 语句出错或没有返回正确的结果

这种情况经常在使用动态 SQL 语句时出现，以下代码就存在一个错误。

```
<?php
    mysqli_connect("localhost", "root", "111") or die;
    mysqli_select_db("db_database17 ");
    $ sql = "select * from $ table";
    $ result = mysqli_query( $ sql);
    print_r(mysqli_fetch_row( $ result));
?>
```

上述代码中错误地使用了一个没有赋值的变量 $ table 作为操作的数据表名称，结果返回如下错误信息：

```
Warning: mysqli_fetch_row(): supplied argument is not a valid MySQL result resource in E:\wamp\
www\index.php on line 6
```

解决方案：使用 print 或者 echo 函数输出 SQL 语句来检查错误。例如对上述代码进行修改，通过 echo 语句直接输出 $ sql 的值，查看这个 SQL 语句是否正确，其代码如下：

```
<?php
    mysqli_connect("localhost", "root", "123456") or die;  //连接数据库
    mysqli_select_db("db_database17 ");                    //选择数据库
    $ sql = "select * from $ table";                       //定义 SQL 语句
    echo $ sql;                                            //输出 SQL 语句
    $ result = mysqli_query( $ sql);                       //执行 SQL 语句
```

```
    print_r(mysqli_fetch_row( $ result));                    //输出执行结果
?>
```

如此,从运行结果就可以看出 SQL 语句中的错误了。

5. 数据库乱码问题

在获取数据库中的数据时,中文字符串的输出出现乱码。

(1) 问题分析。输出数据库中的数据之所以会出现乱码,是因为在获取数据库中的数据时,数据本身所使用的编码格式与当前页面的编码格式不符,从而导致输出数据乱码。

(2) 解决方案。在与 MySQL 服务器和指定数据库建立连接后,应用 mysqli_query()函数设置数据库中字符的编码格式,使其与页面中的编码格式一致。

```
<?php
 $ conn = mysqli_connect("localhost","root","123456");        //连接数据库服务器
mysqli_select_db("db_database17", $ conn);                    //连接 db_database17 数据库
mysqli_query("set names uft8");                               //设置数据库编码格式
?>
```

上述通过 mysqli_query()函数设置的编码格式是 uft8,同样还可以设置其他编码格式,唯一的一个条件就是要与数据库中的编码格式相匹配。

这就是解决数据库中中文输出乱码的方法,应用 mysqli_query 函数设置数据库的编码格式,使其与页面中的编码格式保持一致,也就不会出现乱码的问题了。

6. 应用 mysqli_error()语句输出错误信息

在执行 MySQL 语句时产生的错误是很难发现的,因为在 PHP 脚本中执行一个 MySQL 的添加、查询、删除语句时,如果是 MySQL 语句本身的错误,程序中不会输出任何的信息,除非对 MySQL 语句的执行进行判断,成功输出什么,失败输出什么。

解决方案:为了查找出 MySQL 语句执行中的错误,可以通过 mysqli_error()语句来对 SQL 语句进行判断,如果存在错误则返回错误信息,否则没有输出,该语句的应用被放置于 mysqli_query()函数之后。

例如,在下面的代码中,在通过 mysqli_query()函数执行查询语句之后,应用 mysqli_error()函数获取 SQL 语句中的错误。

```
<?php
   $ sql = "select  *  from student";                         //定义查询语句,""内少"; "
   $ query = mysqli_query( $ sql, $ conn); //执行查询操作
  echo mysqli_error();                                        //获取 SQL 语句中的错误
      while( $ myrow = mysqli_fetch_array( $ query)){         //循环输出查询结果
?>
```

此方法不仅对查询语句的执行有效,而且对添加、更新和删除语句都适用。是一个查找 SQL 语句本身错误的好方法。

14.5 小　　结

本章介绍了 PHP 访问 MySQL 数据库的方法。重点介绍 PHP 使用 mysqli 接口连接 MySQL 数据库,还讲解了 PHP 中执行 select 语句、insert 语句、update 语句、delete 语句的

方法。学习本章后需要掌握如下内容：

- 利用 PHP 一次执行多个 select 语句的过程。
- PHP 执行多个 select 语句需要使用 multi_query()函数的用法。
- PHP 操作 MySQL 数据库的步骤。
- 通过 MySQLi 函数操作 MySQL 数据库的方法。
- 使用 PHP 操作 MySQL 数据库的基本操作。

习　题　14

1. 思考题

(1) 简述 PHP 语言的基本特点。

(2) 简述利用 PHP 与 MySQL 数据库连接的步骤。

(3) 设置结果集数组的表述方式中的 mysqli_assoc、mysqli_num 和 mysqli_both 分别表示什么？

(4) 简述 mysqli_query()函数的作用。

2. 上机练习题(本题利用 teaching 数据库中的表进行操作)

(1) 利用 PHP 语言查询数据表 course 中的数据。

(2) 利用 mysqli_query()函数向 course 表插入数据。

(3) 利用 mysqli_query()函数删除 course 表数据。

(4) 利用 mysqli_query()函数更新 course 表数据。

(5) 查询教师号为 t05001 的教师信息，并利用输出命令 echo 和 print_r()函数两种方式输出。

第 15 章 基于 JSP 技术的 MySQL 数据库应用开发实例

MySQL 数据库的应用非常广泛，很多的网站和管理系统都使用 MySQL 数据库存储数据。JSP(Java Server Pages)是一种动态网页技术标准。JSP 技术是在传统的网页 HTML 文件中插入 Java 程序段和 JSP 标记从而形成 JSP 文件。JSP 网页的很多技术例如 Hibernate、Spring、Struts 等都是建立在 Java 语言的基础上的。

本章主要介绍如何对在线考试系统的数据库进行设计，并在数据库设计的基础上，应用 NetBeans 集成开发环境，通过 JSP 技术实现一个在线考试系统的设计与开发工作，从而体现 MySQL 数据库在实际应用系统开发中的强大功能。

15.1 实例开发的背景和意义

在线考试系统将传统的考试与网络模式相结合，使教师用户可根据课程自身的特点快速构建考试、测试、学习、调查及分析于一体的网络化考试平台。

15.1.1 项目开发的背景

开发在线考试系统的目的是提高教师的工作效率，让学生实现对所学课程的自我评定，提高学生的自我学习意识和自我管理能力。应用在线考试系统在可预见的未来较长的一段时间内，应该是一种能够改善考试环境、提高教学管理效率、实现课程管理的信息化的必要手段。

在系统的实现过程中，应该结合软件工程的思想，了解当前一般高校的课程考试规范过程，然后从经济、技术、法律和方案等几方面进行可行性分析。从而进行概要设计和逻辑设计。限于资源的关系，目前此项目在进行调研的基础上，首先实现初步的开发，即能够实现对客观题的出题、答题等功能。

15.1.2 系统开发的可行性分析

当系统开发人员接受开发任务时，首先要研究开发任务，判断是否有简单明确的解决办法。事实上，许多问题不可能在一定的系统规模之内解决，如果问题没有可行的解决办法，那么花费在这项开发工程上的任何时间、资源、经费都是无谓的浪费。

可行性研究的目的就是付出较低的开发成本而取得较好的软件功能和较低的软件维护费用，在有限的时间内确定问题是否能够解决。当然，可行性研究的目的不仅仅是要解决问题，还要确定问题是否有研究或经济价值。这就需要利用现有的手段去进行客观分析，在分析权衡几种主要可能方案的利弊的基础上，选择合理的方法与步骤。一般来说，至少应该从

下述几个方面研究每种解法的可行性。

(1) 经济可行性。即进行成本与效益的核算分析,从经济角度判断开发该系统的预期经济效益能否超过它的开发成本。从经济方面来看,基于 JSP 技术的在线考试系统,所用的开发工具和软件基本上都是免费的,在经济上是完全可行的。

(2) 技术可行性。即进行技术风险评估。从开发者的技术实力、工作基础及问题的复杂程度等几方面判断系统开发在时间、费用等限制条件下,利用现有的技术能否实现系统的功能要求以及系统的操作方式是否在某些用户组织内行得通。

本章实例"基于 JSP 的在线考试系统"就是采用"JSP 技术＋MySQL 数据库"组合进行开发的,整个系统是在 NetBeans 8.2 集成环境下进行编写、编译、调试和部署的,其中,Web 服务器使用 Apache-Tomcat-7.0。NetBeans 可以非常方便地安装于多种操作系统平台,包括 Windows、Linux、Mac OS 和 Solaris 等操作系统。提供了强大的 JavaScript 编辑功能,支持使用 Spring 的 Web 框架,并加强了与 MySQL 的整合,NetBeans 可比较方便地使用 MySQL 数据库,启动速度提升幅度很大。在建立一个大工程时,有较低的内存消耗和更快的响应速度。

从技术方面来看,对于 MySQL 数据库与 JSP 技术的结合,在实际应用中是较为成功的解决方案。对于机器本身没有太大的要求,一般个人的电脑完全满足对技术的要求。基于 JSP 的在线考试系统采用了三层体系结构:用户界面层、事物层、数据库层,在用户机上几乎不需要安装任何相关的应用程序,这样不仅应用起来对用户更加方便也同时保证了安全性。同时,JSP 是一种服务器端 HTML 嵌入 Java 代码的脚本语言,是开发动态 Web 网站的很好的工具,在保证很好的操作性的同时,比其他的脚本语言具有更快的执行速度。

如图 15-1 所示,本系统的支持平台具体分为:数据存储层、用户界面层、业务逻辑层。这就能够确保数据的安全性、系统的稳定性和系统的响应速度要求。

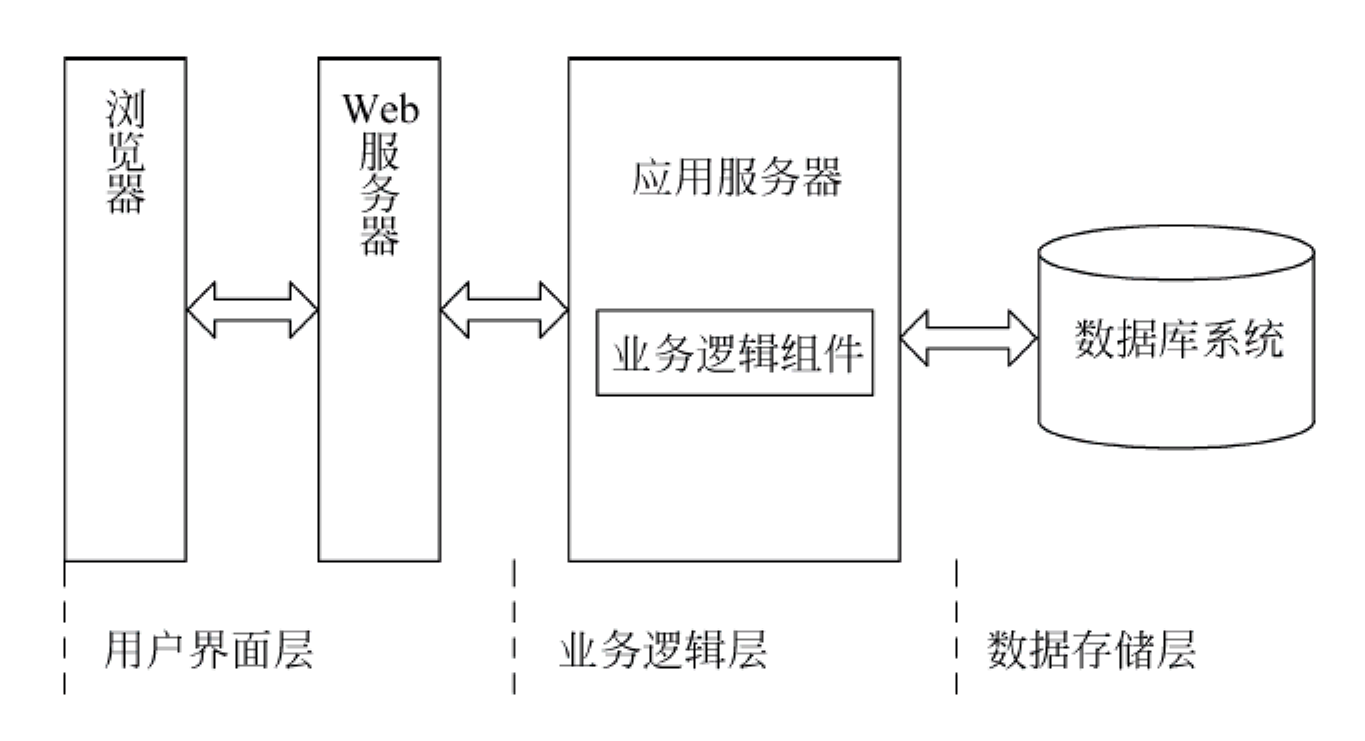


图 15-1　网站应用程序架构

如图 15-2 所示,系统采用 B/S 结构(Browser/Server,浏览器/服务器模式),Web 浏览器是客户端最主要的应用软件。三层 B/S 体系结构可以从技术上保证实现简化客户端操作、实现集中管理与维护的跨平台操作。

在进行实际的软件开发环境搭建时,系统选择应用最为广泛的 Windows 操作系统,Web 服务器端采用 Tomcat＋JSP＋MySQL 的方式,这是在实际应用中已经证明的成功的技术解决方案。而采用 JSP 技术会使得软件运行具有更快的执行速度。

(3) 法律可行性。确定系统开发可能导致的任何知识产权方面的侵权行为和妨碍性后

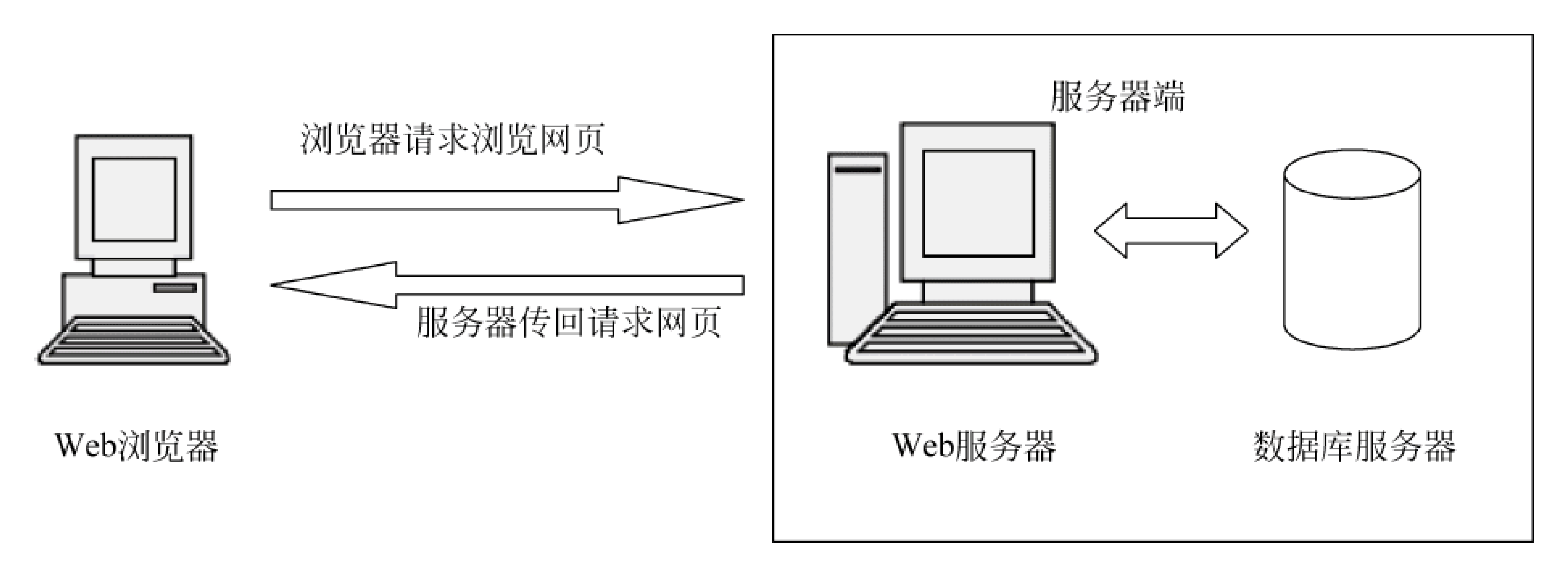


图 15-2　三层 B/S 体系结构

果和责任。

(4) 方案可行性。评价系统或产品开发的几种方案，并进行系统分解，定义各个子系统的功能、性能和界面，最后得出结论性意见。

分析人员应该为每个可行的解法制订一个粗略的实现进度。当然，可行性研究的根本任务是对后续的阶段提出建议，如果问题没有可行解，分析人员应该建议停止这项开发工程，以避免资源浪费；如果问题可行，分析人员应该推荐一个较好的解决方案，并为工程制订一个初步的计划。可行性研究需要的时间长短取决于工程的规模。一般来说，可行性研究的成本只是预期工程总成本的 5%～10%。

15.1.3　开发项目的目标

本系统是基于 JSP 的在线考试系统，主要完成了学生考试、系统改卷、老师增删改查学生成绩及学生信息管理等基本功能，能够基本实现客观题的答题和改卷的功能。由于当前资源的限制，还没能够实现与传统模式下考试完全一样。

本系统很好地解决了客观题考核与改卷的问题，使得老师能够简化劳动，提高工作效率，达到简化考试流程的效果。结合传统的考试方法，项目应该具有的功能如下：

- 基于 JSP 的在线考试系统根据用户类型可划分为系统管理员、教师、学生 3 种类型，更为具体一些就是他们的 id、姓名等。
- 本系统的使用用户有教师、学生。不同的用户具有不同的职责和权限：老师可以进行试题库的更新等操作；学生只负责答题和查看自己的成绩，无法行使教师权限。
- 试卷由系统自动生成，无须人工干预，并保证在试题库足够大的前提下所生成的试题没有重复。

此外，用户操作界面是否好用美观、是否具有人性化设计、是否达到国家或者软件公司的行业标准、软件是否具有很好的兼容性和稳定性等也是评价软件的一些重要指标。

15.2　在线考试系统的数据库设计

按照数据库规范化设计的方法，数据库设计可分为需求分析、概念结构设计、逻辑结构设计、物理结构设计、数据库实施和数据库运行与维护 6 个阶段。在实际的项目开发中，如果系统的数据关系较复杂，数据存储量较大，设计的表较多，表和表之间的关系比较复杂，就

需要首先考虑规范的数据库设计，然后再进行具体的创建库、创建表的工作。所以，数据库设计的重要性不言而喻。

15.2.1 需求分析

了解需求分析的任务，掌握常用的需求分析的方法，可根据不同的应用程序选择不同的需求分析方法进行需求分析。

明确地把需求收集和分析作为数据库设计的第 1 阶段是十分重要的，这一阶段收集到的基础数据(用数据字典来表达)是下一步进行概念结构设计的基础。

数据库设计是指对于一个给定的应用环境，构造优化的数据模型，并据此建立数据库及其应用系统，使之能够有效地存储和管理数据，满足各种用户的应用需求。数据库设计是在 DBMS 支持下进行的，它包括数据库的结构设计和数据库的行为设计。数据库的结构设计是模式与子模式的设计，是信息系统数据模型的静态模型；数据库的行为设计是应用程序设计，是在模型上的动态操作。将数据库的结构设计和行为设计相结合是现代数据库设计的特点之一。

需求分析和概念结构设计阶段是面向现实世界或用户的应用需求，与 DBMS 无关；逻辑结构设计和物理结构设计阶段是面向 DBMS 的；数据库实施和数据库运行与维护阶段面向“实现”。

每个设计阶段完成后要根据一定的指标对设计结果进行评价，对不满足用户要求的部分进行分析和修改，所以数据库设计是一个不断反复、逐步完善的过程。

1. 需求分析的基本过程

常用的需求分析方法有调查客户的公司组织情况、各部门的业务需求情况、协助客户分析系统并请用户填写，查阅业务相关数据记录等。

需求是用户要求数据库应用系统必须满足的所有功能和限制，它包括功能要求、性能要求、可靠性要求、安全性和完整性要求等限制，其中功能要求又包括信息要求和处理要求。需求分析就是通过与用户的沟通和交流获取用户的需求，并对需求进行分析和整理，最终形成需求文档。需求分析包括需求获取、需求分析和处理等多个过程。

需求分析是数据库设计的首要任务，也是后续设计工作的基础。通过调查，详细了解用户的每一个业务过程和业务活动的工作流程及信息处理流程，准确理解用户对信息系统的需求，使需求分析尽可能充分与准确。需求分析的重点是调查、收集并分析客户业务的数据需求、处理需求、安全性和完整性需求。

需求分析的任务是通过详细调查现实世界要处理的对象，充分了解原系统工作概况，明确用户的各种需求，然后在此基础上确定新系统的功能。新系统必须充分考虑今后可能的扩充和改变，不能仅仅按当前应用需求来设计数据库。

调查的重点是“数据”和“处理”，通过调查、收集与分析，获得用户对数据库的如下要求：

(1) 信息要求：指用户需要从数据库中获得信息的内容与性质，由信息要求可以导出数据要求，即在数据库中需要存储哪些数据。

(2) 处理要求：指用户要完成什么处理功能，对处理的响应时间有什么要求，处理方式是批处理还是联机处理。

(3) 安全性与完整性要求：对所有系统用户需要有完善的口令加密功能，以保证系统

及数据的安全性。应用软件对输入的数据进行合法性、有效性和完整性检验，如果输入数据存在问题，系统应能及时给予提示。

2. 获取需求的内容

通过调查来获取用户的实际需求，采用的调查方法有开调查会、用户访谈、问卷调查法和参加业务实践等，针对不同用户采用不同的调查方法，一般是几种方法互补使用。需要调查的内容有：

(1) 调查组织结构。要建立数据库应用系统，首先要清楚当前系统的组织结构情况，即了解该组织各部门的划分及其相互关系、各部门的职责、人员配备、业务分工等。调查结果可用组织结构图来描述。

(2) 调查管理功能。该功能指的是完成某项工作的能力。每个系统都有一个总目标，为了达到总目标，必须完成各个子系统的功能，子系统的功能又依赖于其下面各项更具体功能的实现。在调查中，可以用功能层次图来描述从系统目标到各项功能的层次关系。

(3) 调查各部门的业务流程。调查各部门的处理业务、信息来源、处理方法、计算方法、信息流经去向、提供信息的时间和形态以及安全性和完整性要求，调查结果用业务流程图来描述。

(4) 确定新系统的边界。一个组织业务活动的管理不可能全部由计算机来完成，所以设计人员通过对上述调查结果的分析来确定系统的边界，即确定哪些功能由计算机完成或将来准备让计算机完成，哪些活动由人工完成。由计算机完成的功能就是新系统要实现的功能。

3. 需求的处理

按照某种分析方法对所获得的需求进行分析，典型的分析方法有结构化分析方法和面向对象分析方法。结构化分析方法是一种面向过程的方法，它以过程为中心建立系统用户需求模型，常用的分析工具主要有数据字典、数据流程图等。面向对象分析方法就是运用面向对象的方法，对问题域和系统责任进行分析和理解，正确认识其中的事物和它们之间的关系，找出描述问题域和系统责任所需的类及对象，定义这些类和对象的属性和服务，以及它们之间所形成的结构、静态联系和动态联系，并产生面向对象的模型。

(1) 分析业务流程。了解某项业务的具体处理过程，发现和处理系统调查工作中的错误和疏漏，修改和删除原系统的不合理部分，在新系统基础上优化业务处理流程。

(2) 分析系统数据。在调查的基础上，进一步收集和分析数据，主要包括：

- 明确用户在数据库中需要存储哪些数据？即确定各实体以及各实体集所包含的属性。
- 明确各实体集之间的联系，即确定联系的类型。
- 明确各属性的组成，即属性的名称、类型、长度、值域、使用特点等。

15.2.2 数据字典的开发

数据字典主要是对数据项、数据结构和数据存储等几方面进行具体的定义。

(1) 数据项。数据项又称为数据元素，是数据的最小单位，描述数据的静态特性，其定义包含如下内容：

数据项的描述＝{数据项名称，别名，描述，数据类型及取值长度，取值范围，取值含义，

存储处}

例如,“学号”数据项的定义如下:

① 数据项名称:学号(studentID)。

② 别名:学生编号。

③ 描述:学号是学生信息表的主码,每个学生都有一个唯一的学号。

④ 数据类型及取值长度:字符型,6~20 位。

⑤ 取值范围:6~20 位数字字符。

⑥ 取值含义:学号编码可以有一定的规则,比如 2 位年,1 位性别,2 位学院,3 位专业,3 位专业排名。

⑦ 存储处:学生表 tb_student。

(2) 数据结构。数据结构描述某些数据项之间的关系。一个数据结构可以由若干个数据项组成,也可以由若干个数据结构组成,还可以由若干个数据项和数据结构组成。其定义包含如下内容:

数据结构的描述={数据结构名称,描述,数据结构组成,其他说明}

例如,“试题类型表”数据结构的定义如下:

① 数据结构名称:试题类型表 tb_type。

② 描述:包括试题类型的主要信息。

③ 数据结构组成:编号+试卷名称+试卷类型描述。

④ 其他说明:在系统功能扩充时可能增加定义项。

(3) 数据存储。数据存储在数据字典中只描述数据的逻辑存储结构,而不涉及它的物理组织,其定义包含如下内容:

数据存储的描述={数据存储名称,描述,数据存储组成,主码,相关联的处理}

例如,“成绩”数据存储的定义如下:

① 数据存储名称:成绩。

② 描述:存放学生某门课程的成绩。

③ 数据存储组成:学生的某门课程的成绩,即由编号+学生编号+对应学科类型+考试科目类型+考试成绩+学生姓名组成。

④ 主码:编号。

15.2.3 设计数据库的概念结构

学会将现实世界的事物和特性抽象为信息世界的实体间的联系,能够使用实体联系图(E-R 图)描述实体、属性和实体之间的联系。

概念结构设计是在需求分析的基础上,形成一个反映用户信息需求的并且独立于计算机硬件和 DBMS 的概念结构。将在线考试系统需求分析得到的用户需求抽象为信息结构即概念模型的过程就是概念结构设计,它是整个数据库设计的关键,本项目要求把在线考试系统抽象出来,绘制出在线考试系统的 E-R 图。

若要将现实世界中的事物直接存储到计算机中进行处理,就必须能够在需求分析的基础上把得到的用户需求抽象出来,对它们进行数据化后才能存储到计算机中进行处理。本项目以在线考试系统为具体应用,介绍如何将现实世界的客观事物进行数据化,然后绘制

E-R 图。

1. 数据模型和概念模型

在数据库技术中，数据是数据库中存储的基本对象。数据是信息的载体，信息是一种已经被加工为特定形式的数据。

(1) 数据模型。数据模型就是对现实世界数据的模拟和抽象，而数据库要基于某种数据模型组织和存储数据，数据模型是严格定义的一组概念的集合，这些概念精确地描述了系统的静态特性、动态特性和完整性约束条件(Integrity Constraints)，因此数据模型通常由数据结构、数据操作和完整性约束 3 部分组成。

(2) 概念模型。概念模型用于信息世界的建模，是现实世界到信息世界的第一层抽象，是数据库设计人员进行数据库设计的有力工具，也是数据库设计人员和用户之间进行交流的语言，因此概念模型一方面应该具有较强的语义表达能力，能够方便、直接地表达应用中的各种语义知识，另一方面它还应该简单、清晰、易于用户理解。

概念模型包括实体、属性码、域、实体型、实体集和联系(Relationship)等元素。在现实世界中，事物内部以及事物之间都是有联系的，这些联系在概念模型中反映为实体(型)内部的联系和实体(型)之间的联系。实体内部的联系通常是指组成实体的各属性之间的联系；实体之间的联系通常是指不同实体集之间的联系。

(3) 实体之间的联系。实体之间的联系可以归纳为 3 种类型，即一对一联系(1∶1)、一对多联系(1∶n)和多对多联系(m∶n)。

2. 概念模型的描述方法

目前描述概念模型的最常用方法是“实体-联系”(Entity-Relationship，E-R 图)方法，E-R 图中包括实体、属性和联系 3 种图素。

实体用矩形框来表示，属性用椭圆形框来表示，联系用菱形框来表示，框内填入相应的实体名和联系名；实体与属性或者实体与联系之间用直线连接。E-R 图中使用的基本符号如图 15-3 所示。

(1) 绘制“在线考试系统”各个实体图。针对“在线考试系统”的需求，抽取出各实体及其所需属性形成各个实体图。

学生实体图如图 15-4 所示。

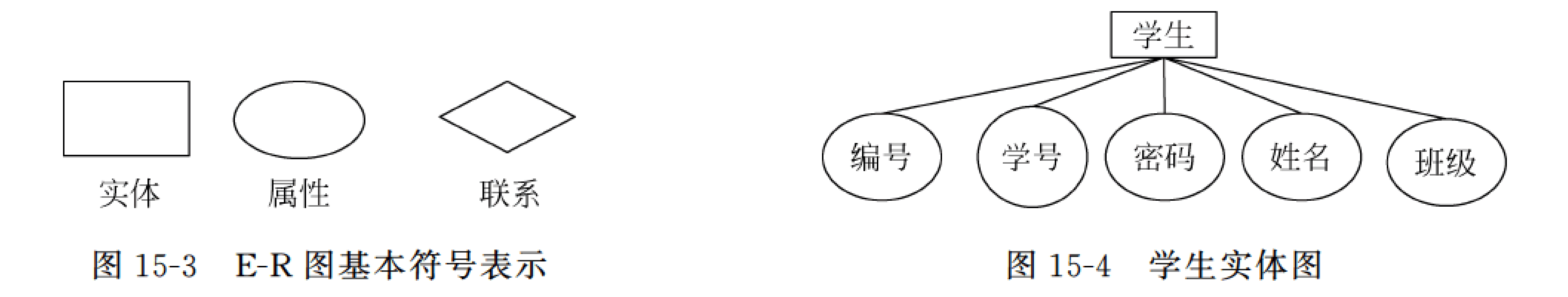


图 15-3　E-R 图基本符号表示

图 15-4　学生实体图

题目实体如图 15-5 所示。

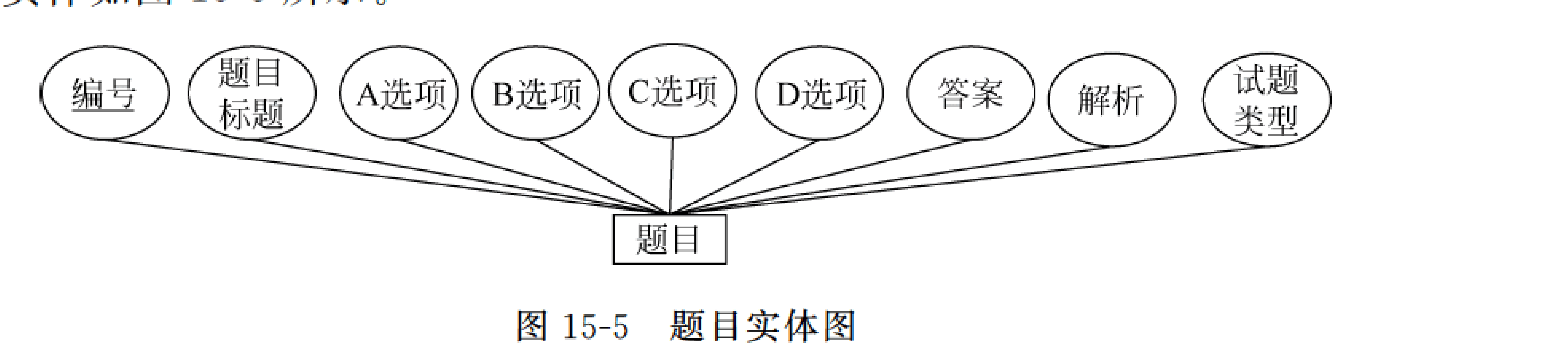


图 15-5　题目实体图

试题类型实体图如图 15-6 所示。

教师在系统中兼任管理员，教师实体中简化设置教师号和密码，实体图如图 15-7 所示。



图 15-6　试题类型实体图

图 15-7　教师实体图

（2）绘制"在线考试系统"全局 E-R 图。对各局部 E-R 图汇总后得到整个"在线考试系统"的全局 E-R 图，如图 15-8 所示。教师能够对学生进行添加、删除、修改操作，能够对试题类型（或考试科目）进行添加删除及修改，并能够实现对试题题目的添加删除和修改。学生能够通过登录本系统选择试题类型，系统能够自动生成试卷，学生考试后系统自动判卷出成绩。另外，教师还能够实现若干查询操作。

图 15-8　在线考试系统全局 E-R 图

在需求分析阶段所得到的应用需求应该是信息世界的结构的抽象，概念设计阶段则是需要将这些抽象信息利用某种方式表达出来，E-R 图是这些表达方式之一。只有这样才能

更好、更准确地在下一步利用 MySQL 实现这些需求。

15.2.4 设计数据库的逻辑结构

本阶段的任务就是能够将概念结构设计阶段绘制的 E-R 图转换为关系模式，根据开发需求，将关系模式规范化到一定的程度。

逻辑结构设计是将概念结构转换为 MySQL 所支持的数据模型，并对数据模型进行优化。E-R 图表示的概念模型是直接表达用户的各种需求的，它独立于任何一种数据模型，与计算机硬件无关，与 DBMS 无关。而逻辑结构设计就是将概念结构设计阶段完成的 E-R 模型转换为所选择的 DBMS 所支持的数据模型，并对数据模型进行优化。

关系模型是目前数据库系统普遍采用的数据模型，也是应用最广泛的数据模型，关系模型通过二维表来表示实体以及实体之间的联系，本项目将详细介绍关系模型以及如何将 E-R 模型转换为关系模型。

1. 数据的组织方式

数据库中的数据是按照一定的逻辑结构存储的，这种结构是用数据模型来表示的。现有的数据库管理系统都是基于某种数据模型的，按照数据库中数据采用的不同联系方式，数据模型可以分为 3 种：层次模型、网状模型和关系模型。

关系模型与层次和网状模型的理论和风格截然不同，如果层次和网状模型是用“图”表示实体和实体之间的联系，那么关系模型则是用“二维表(关系)”来表示实体和实体之间的联系的。从现实世界中抽象出的实体及实体之间的联系都使用关系这种二维表来表示。而关系模型就是用若干个二维表来表示实体及实体之间的联系，这是关系模型的本质，学生关系模型如表 15-1 所示。

表 15-1 学生关系模型

学号	姓名	成绩	班级	流水编号	密码
123456789	招望舒	95	软件 1504	2	123456
123457899	李明华	90	计科 1502	3	123456
070408000	徐 良	85	软件 1505	4	123456
081523000	张思睿	80	软件 1502	5	123456
091578900	王法务	90	软件 1502	6	123456
181234567	徐赛文	98	软件 1505	7	123456
123456798	苏西坡	95	软件 1504	8	123456

2. 关系模型的基本概念

从用户观点看，关系模型由一组关系组成，每个关系的数据结构是一张规范化的二维表。在关系数据库中，有以下几个常见的关系术语。

(1) 关系。关系就是一个二维表格，每个关系都有一个关系名，在 MySQL 中的关系称为表(Table)。

(2) 元组(记录)。在一个具体的关系中，表中的一行称为一个元组(记录)。

(3) 属性(字段)。表中的一列即为一个属性(字段)，给每一个属性起一个名称即属性名。

(4) 域。属性的取值范围,如成绩一般的范围是 0～100,性别的域是(男,女)等。

(5) 码。也称为键。表中的某个属性组,它可以唯一确定一个元组,如学生学号,可以唯一确定一个具体的学生,也就成为学生关系的码。

(6) 分量。元组中的一个具体的属性值,称为分量。

(7) 关系模式。对关系的描述称为关系模式,一个关系模式对应一个关系,是命名的属性集合,一般表示为:

```
关系名(属性名 1,属性名 2,…,属性名 n)
```

例如:

```
学生(学号,姓名,成绩,班级名称,流水编号,密码)
```

3. 关系的基本性质

关系表现为二维表,但不是所有的二维表都是关系。基本关系具有如下 6 条性质:

(1) 列是同质的(Homogeneous),即每一列中的分量是同一类型的数据,来自同一个域。

(2) 不同的列可出自同一个域,称其中的每一列为一个属性,不同的属性要给予不同的属性名。

(3) 列的顺序无所谓,即列的次序可以任意交换。

(4) 任意两个元组的候选码不能相同。

(5) 行的顺序无所谓,即行的次序可以任意交换。

(6) 分量必须取原子值,即每一个分量都必须是不可分的数据项。

关系模型要求关系必须是规范化(Normalization)的,即要求关系必须满足一定的规范条件。这些规范条件中最基本的一条就是:关系的每一个分量必须是一个不可分的数据项。

4. 数据规范化

(1) 数据依赖是一个关系内部属性与属性之间的一种约束关系。这种约束关系是通过属性间值的相等与否体现出来的数据间的相关联系,它是现实世界属性间相互联系的抽象,是数据内在的性质,是语义的体现。比如描述一个学生的关系,可以有学号、姓名、性别、班级编号等几个属性。由于一个学号只对应一个学生,所以一旦"学号"值确定后,学生的姓名、性别、班级编号等的值也就被唯一地确定了。

例如:建立一个描述学校考务系统的数据库,该数据库涉及的对象包括学生学号、学生姓名、班级编号、班级名称、课程名和成绩。假设用一个单一的关系模式来表示,则该关系模式的属性集合为:

U={学生学号,学生姓名,班级编号,班级名称,课程名,成绩}

考察这个关系模式发现存在以下问题:

① 数据冗余度大:班级名称重复出现,重复次数与该班所有学生的所有课程成绩出现次数相同。

② 更新异常:由于数据冗余,当更新数据库中的数据时,系统要付出很大的代价来维护数据库的完整性,否则会面临数据不一致的危险。

③ 插入异常:如果一个班级刚刚成立,尚无学生选课记录,则系统无法把该班级信息存入数据库。

④ 删除异常：如果某个班级的学生全部毕业了，在删除该班学生信息的同时，把这个班的信息也一起删除掉了。

鉴于存在以上种种问题，可以得出结论：学生关系模式不能满足不会发生插入异常、删除异常、更新异常以及数据冗余度应尽可能的小等规则。

(2) 规范化设计。在数据库设计时，有一些专门的规则，称为数据库的设计范式，遵守这些规则，将创建设计良好的数据库。

在实际的数据库设计过程中，在利用规范化设计考察关系模式时，可以针对不同的关系以及关系转化成的表可以预估的数据量、物理存取路径等因素，对规范化关系采用一些另外的处理方法。

在本系统中，优化数据模型需要对 MySQL 适合的模型进行转换。转换的主要依据是 MySQL 数据库管理系统的功能及限制。

因此，数据库逻辑设计的结果不是唯一的。得到初步数据模型后，还应该适当地修改、调整数据模型的结构，以进一步提高 MySQL 数据库应用系统的性能，这就是数据模型的优化。

关系数据模型的优化通常以规范化理论为指导。优化数据模型的方法主要体现在如下几个方面：

- 确定数据依赖。按需求分析阶段所得到的语义，分别写出每个关系模式内部各属性之间的数据依赖以及不同关系模式属性之间的数据依赖。
- 对于各个关系模式之间的数据依赖进行极小化处理，消除冗余的联系。
- 按照数据依赖的理论对关系模式进行分析，考察是否存在部分函数依赖、传递函数依赖、多值依赖等，确定各关系模式分别属于第几范式。
- 按照需求分析阶段得到的各种应用对数据处理的要求，分析对于这样的应用环境这些模式是否合适，确定是否要对它们进行合并或分解。

并不是规范化程度越高的关系就越优，当查询经常涉及两个或多个关系模式的属性时，系统必须经常地进行连接运算。连接运算的代价是相当高的，因此在这种情况下，第二范式甚至第一范式也许是适合的。非第三范式规范化关系模式虽然会存在不同程度的更新异常，但如果在实际应用中对此关系模式只是查询，并不执行更新操作，就不会产生实际影响。对于一个具体应用来说，到底规范化进行到什么程度，需要权衡响应时间和潜在问题两者的利弊才能决定。

5. E-R 模型向关系模型的转换过程

E-R 模型向关系模型的转换主要解决两个问题，一个是如何将实体型和实体间的联系转换为关系模式，另一个是如何确定这些关系模式的属性组成和码。因为关系模型的逻辑结构是一组关系模式的集合，而 E-R 模型则是由实体型、属性和实体之间的联系组成的，所以将 E-R 模型转换为关系模型就是将实体型、属性和实体之间的联系转换为关系模式。

具体的通用转换原则如下：

(1) 实体型转换为一个关系模式，实体的属性就是关系的属性，实体的码就是关系的码。

(2) 一个 1∶1 的联系，可以单独转换为一个独立的关系模式，也可以与任意一端对应的关系模式合并。若转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系

本身的属性均转换为关系的属性，每个实体的码均是该关系模式的候选码。若与任意一端对应的关系模式合并，则需要在该关系的属性中加入另一端关系模式的码和联系本身的属性。

(3) 一个 1：n 的联系，可以单独转换为一个独立的关系模式，也可以与 n 端对应的关系模式合并。若转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，该关系的码为 n 端实体的码。若与 n 端对应的关系模式合并，则与该关系相连的一端实体的码以及联系本身的属性需要加入 n 端关系模式中，n 端关系模式的码为该关系模式的码。

(4) 一个 m：n 的联系只能单独转换为一个独立的关系模式。与该联系相连的各实体的码以及联系本身拥有的属性均转换为关系的属性，该关系的码为两端实体码的组合。

(5) 3 个或 3 个以上间的一个多元联系只能转换为一个独立的关系模式。与该多元联系相连的各实体的码以及联系本身拥有的属性均转换为关系的属性，该关系的码为各实体码的组合。

(6) 具有相同码的关系模式可以合并。

概念结构是独立于任何一种数据模型的信息结构，逻辑结构设计的任务就是将概念结构设计阶段设计好的基本 E-R 图转换为与选用的 DBMS 产品所支持的数据模型相符合的逻辑结构，为下一步的数据库应用程序开发提供逻辑模式或外模式，并能够在 MySQL 数据库服务器上构建数据库，基于该数据库构造数据表，完成数据表中数据的基本操作，并能够应用某种开发工具完成数据库应用程序开发。

15.2.5 设计数据表

数据库设计时要确定创建哪些表、表中有哪些字段、字段的数据类型和长度。本章介绍的在线考试管理系统选择 MySQL 数据库。因为本书主要是介绍 MySQL 数据库的知识，所以在设计数据库时会尽量用到书中介绍过的 MySQL 数据库的知识点。这样可以让读者对 MySQL 数据库有一个全面的认识。

本系统由于功能相对简单，所有的表都放 examsystem 数据库下，创建 examsystem 数据库后，在该数据库下一共存放 5 张表，分别是学生表 tb_student、试题类型表 tb_type、题目表 tb_subject、成绩表 tb_score 和教师表 tb_teacher。其中，tb_teacher 表中存储管理员的用户名和密码；tb_student 表存储学员的信息；tb_type 表存储课程类型信息；tb_subject 表存储试题，相当于题库；tb_score 表存储学生的成绩信息。

索引是创建在表上的，是对数据库表中一列或多列的值进行排序的一种结构。

数据库 examsystem 中的 5 张表结构如表 15-2～表 15-6 所示。

表 15-2　tb_student 表结构

字　段　名	数据类型	是否主键	描　　述
id	varchar	是	主键，流水号
studentID	varchar	否	学生编号
password	varchar	否	密码
studentName	varchar	否	姓名
sclass	varchar	否	班级

表 15-3　tb_type 表结构

字　段　名	数据类型	是否主键	描　　述
id	int	是	编号
name	varchar	否	试卷名称
description	varchar	否	试卷类型描述

表 15-4　tb_subject 表

字　段　名	数据类型	是否主键	描　　述
subjectID	int	是	题目编号
subjectTitle	varchar	否	题目标题
subjectOptionA	varchar	否	A 选项
subjectOptionB	varchar	否	B 选项
subjectOptionC	varchar	否	C 选项
subjectOptionD	varchar	否	D 选项
subjectAnswer	varchar	否	答案
subjectParse	text	否	解析
subjectType	int	否	试题类型

表 15-5　tb_teacher 表

字段名	数据类型	是否主键	描述
teacherID	varchar	是	教师编号
password	varchar	否	密码

表 15-6　tb_score 表

字　段　名	数据类型	是否主键	描　　述
id	int	是	主键
stuId	int	否	对应学生 id
typeId	int	否	对应科目类型 id
typeName	varchar	否	科目类型名称
result	float	否	考试成绩
stuName	varchar	否	学生姓名

15.3　在线考试系统的应用开发

本节通过该在线考试系统的开发，详细介绍使用 MySQL 进行 JSP 应用程序的开发过程，并按照软件开发方法与流程，从需求分析到编码实施，可以理解软件项目开发的基本过程。

15.3.1　在线考试系统的功能分析

了解可行性分析、需求分析在软件开发过程中的作用，给定任意语义的系统描述，能够分析这些系统的功能需求，并绘制系统需求的 UML 用例图。

完全理解软件需求对于软件开发工作的成功是至关重要的，需求说明的任务是发现、规

范的过程，有益于提高软件开发过程中的能见度，便于对软件开发过程中的控制与管理，便于采用工程方法开发软件，提高软件的质量，便于开发人员、维护人员、管理人员之间的交流、协作，并作为工作成果的原始依据，并且在向潜在用户传递软件功能、性能需求，使其能够判断该软件是否与自己的需求相关。

任何一个软件系统的设计与开发都需要进行详细的需求分析，其目的就是尽量快速、准确、全面地获得系统的真实需求，规划出系统的整体功能，为系统的设计与实现做好完备而坚实的基础，使系统的开发工作得以顺利进行。

1. 系统性能需求

性能需求是指系统必须满足的定时约束或容量约束，通常体现在终端用户接入速率、响应时间、稳定性、可扩展性和并发用户支持等几个方面。因为本系统包含成绩有关的数据操作，所有数据不但要保证100%准确、可靠，而且还要保证100%安全。所以，在现有功能规划的基础上，必须满足以下性能需求：

(1) 可靠性。系统要求选用可靠的计算机及网络设备，数据库服务器等在允许的条件下还需采用磁盘镜像技术。数据库及操作系统软件要采用成熟的、能提供有效技术支持的主流产品，应用软件的设计编制应遵循规范化标准，整个项目的开发过程应得到有效监控。

(2) 安全性。系统的安全性是非常重要的，合理的安全控制可以使应用系统中的信息资源得到有效的保护，故系统在数据库层和应用层都做了安全方面的设置：对所有系统用户需要有完善的口令加密功能，以保证系统及数据的安全性。

(3) 完整性。应用软件对输入的数据进行合法性、有效性和完整性检验，如果输入数据存在问题，系统应能及时给予提示。

(4) 易用性。系统的软硬件设计面向非专业的管理人员，管理系统具有美观友好的操作界面，使用简捷、易懂易学。

2. 分析在线考试系统的功能需求

可行性分析也称为可行性研究，是在考察的基础上，针对新系统的开发是不是具有必要性和可能性，对新系统的开发从经济、技术、法律、方案等方面进行分析和钻研，以避免投资失误，确保新系统的开发顺利。可行性研究的目标便是用最小的代价在尽量短的时间内确定问题是不是能够解决。

在线考试系统不仅仅是对学生成绩的管理，还涵盖与学生成绩紧密相关的教师信息、学生信息和课程信息的管理。

从教务管理流程和管理方便的角度考虑，对系统提出以下要求：

(1) 需要进行身份认证登录。系统只允许合法用户进行登录操作，该系统主要面向三类用户：一类为学生用户；一类为教务人员用户；一类为教师用户。合法用户登录后可以进行系统的主要功能操作。本系统为了简化设计，将教师用户和管理员的功能进行了合并。

(2) 具备数据管理功能。数据管理主要是对班级基本信息、学生基本信息、课程基本信息、教师基本信息、教师讲授班级课程信息、学生选修课程成绩信息、与登录用户相关的用户信息等基础数据进行的管理和维护工作。

(3) 具备教师讲授班级课程管理功能。教师讲授班级课程主要完成对教师讲授哪个班级哪门课程进行设置的操作，把教师、班级和课程之间的讲授信息进行关联。

(4) 具备成绩管理功能。成绩管理主要是对学生在校期间的学习成绩进行录入、修改

和管理。

3. 功能分析

根据需求分析，在线考试系统包括教师信息管理、学生信息管理、题目信息管理、成绩管理、科目管理、数据查询及系统管理等，系统模块结构图如图 15-9 所示。

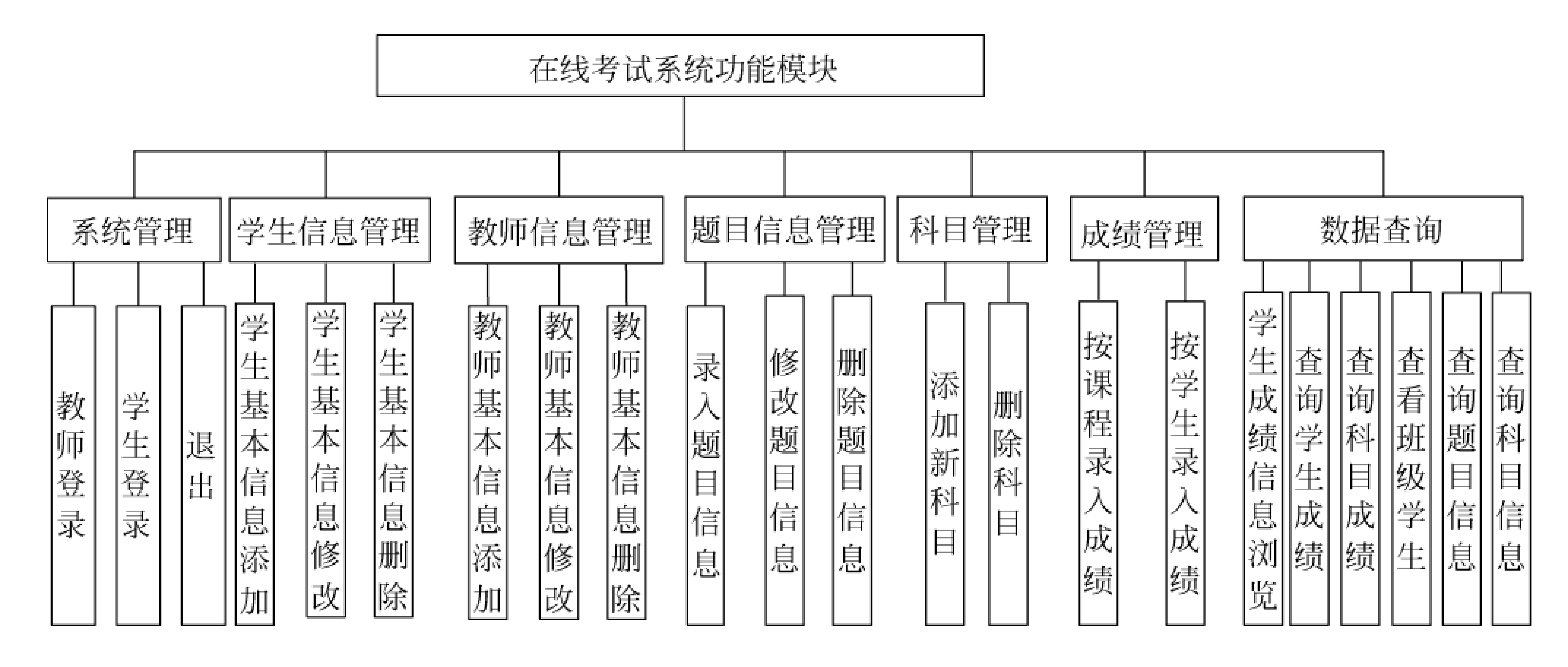


图 15-9　在线考试系统模块结构图

15.3.2　在线考试系统的系统实现

1. 在线考试系统的建设目标

(1) 登录功能，根据用户类型划分为学生角色和教师角色。由于在考试系统中的定位不同，生成的权限因而不同。教师的职责是组织考试的进行并更新后台数据库，学生的职责则是进行考试并在考完之后查看成绩。

(2) 自动生成试卷，只要后台数据库的题库足够庞大，在学生选定考试科目之后，程序会根据难度随机从题库中抽取试题组成试卷，并保证每一位学生拿到的考题不会完全相同。

(3) 阅卷和统计分数，在考生做完题目并单击“提交”按钮之后，系统会自动根据标准答案进行阅卷，阅卷成绩会及时显示给考生，并将成绩传到后台数据库进行保存。

(4) 学生成绩查询，在学生提交试卷后，就能够查看自己的成绩。

(5) 数据库其他操作，对学生信息及考试题目的增删改查。

2. 模块层次结构分析

在线考试系统中，教师、学生、试卷、试题等都能够进行数据的增删改查。

(1) 在线考试：成功登录后，会提示考生进行何种科目的考试，考生在选定考试科目后，会马上进行答题，在规定时间内答完，提交答题信息之后，系统能即刻评卷，将考试成绩显示给考生。

(2) 成绩管理：系统为方便查询，设定 4 种查询方式：查询所有成绩、查询科目成绩、通过姓名查找学生成绩、查找某班级全部学生成绩。

(3) 后台数据库管理：此功能由教师来完成。教师能进行试题管理、学生管理、成绩管理。其中学生管理指学生信息管理(包括姓名、班级等)；试题管理包括录入试题类型、录入试题、管理试题、查询试题。

3. 构建工程

首先，在 NetBeans IDE 8.2 中创建一个 Java Web 工程，并将这个 Java Web 工程取名为 exam。服务器选用 Apache Tomcat 类型，如图 15-10 所示。本工程的所有 JSP 页面都放在 WebRoot 文件夹下。



图 15-10　选择服务器

4. 数据层功能实现

基于 JSP 的在线考试系统的持久化逻辑以 Hibernate 为中间件，采用 DAO 设计模式。DAO 模式是 Java EE 核心模式中的一种，其主要是在业务核心方法和具体数据源之间增加一层，减少耦合性。

每个持久化类对应一个 DAO，它实现了持久化类的创建、查询、更新及删除方法和其他访问持久化机制的方法。在相应的 DAO 实现中，调用 Hibernate API 访问持久层。这样只有特定于 Hibernate 的 DAO 实现需要依赖 Hibernate API，当改用其他的持久化机制或持久化中间件时，只需要创建新的 DAO 实现，无须更改应用中其他业务逻辑代码。

基于 JSP 的在线考试系统以 MySQL 为后台数据库，通过 Hibernate 访问数据库的配置文件 hibernate.cfg.xml 的主要内容如下：

```
    <?xml version = '1.0' encoding = 'UTF - 8'?>
    <!DOCTYPE hibernate - configuration PUBLIC
            " - //Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate - configuration - 3.0.dtd">
    < hibernate - configuration >
    < session - factory >
        < property name = "dialect">
            org.hibernate.dialect.MySQLDialect
        </property ><! -- 数据库方言 -->
```

```
        <property name = "connection.url">
            jdbc:mysql://localhost:3306/db_examsystem
        </property><!-- 数据库连接 URL -->
        <property name = "connection.username"> root </property>
        <!-- 数据库用户名 -->
        <property name = "connection.password"> 123456 </property>
        <!-- 数据库用户密码 -->
        <property
name = "connection.driver_class"><!-- com.mysql.jdbc.Driver -->
            com.mysql.jdbc.Driver
        </property>
        <property name = "myeclipse.connection.profile"> mysql </property>
        <property name = "show_sql"> true </property>
        <mapping resource = "com/sanqing/po/Student.hbm.xml" />
        <mapping resource = "com/sanqing/po/Teacher.hbm.xml" />
        <mapping resource = "com/sanqing/po/Type.hbm.xml" />
        <mapping resource = "com/sanqing/po/Subject.hbm.xml" />
    </session-factory>
    </hibernate-configuration>
```

5. 创建对象/关系映射

根据数据库的各个表创建映射文件。tb_teacher 表、tb_student 表、tb_subject 表、tb_type 表、tb_score 表都对应产生 Hibernate 映射文件。

```
//Student 类的映射文件
<hibernate-mapping>
    <class name = "com.sanqing.po.Student" table = "tb_student"><!-- 每个 class 对应一个持
久化对象 -->
<id name = "studentID" type = "string"><!-- id 元素用来定义主键标识,并指定主键生成策略 -->
<generator class = "assigned"></generator>
</id>
<property name = "password"></property><!-- 映射 password 属性 -->
<property name = "studentName"></property><!-- 映射 studentName 属性 -->
<property name = "result"></property><!-- 映射 result 属性 -->
<property name = "sclass"></property><!-- 映射 sclass 属性 -->
</class>
</hibernate-mapping>
```

6. 创建持久化类

根据创建的映射文件,创建持久化类。基于 JSP 的在线考试系统使用的持久化类有:

Student 类:对应学生实体,实现学生信息的持久化工作。

Teacher 类:对应教师实体,实现教师信息的持久化工作。

Subject 类:对应题目实体,实现题目信息的持久化工作。

Type 类:对应试卷类型实体,实现试卷类型信息的持久化工作。

Score 类:对应学生成绩实体,实现学生成绩信息的持久化工作。

7. 创建 DAO 实现

SubjectDAO 接口定义了系统进行题目管理的方法,包括题目的增删改查等。其接口定义如下所示:

```
public interface SubjectDAO {
    public void addSubject(Subject subject);                //保存方法,用来保存试题
    public Subject findSubjectByTitle(String subjectTitle);
                                                            //根据试题标题查找试题
    public List<Subject> findSubjectByPage(Page page);      //分页查询试题
    public int findSubjectCount();                          //查询试题总量
    public Subject findSubjectByID(int subjectID);          //根据试题 ID 查找试题
    public void updateSubject(Subject subject);             //更新方法,用来更新试题
    public void deleteSubject(int subjectID);               //根据试题 ID 删除试题
    public List<Subject> likeQueryByTitle(String subjectTitle,Page page);
                                                            //根据试题标题模糊查询试题
    public int findLinkQueryCount(String subjectTitle);     //查询模糊记录数
    public List<Subject> randomFindSubject(int number);     //随时取出记录
}
```

15.3.3 系统功能模块的实现

本项目模型由实现业务逻辑的 NetBeans 构成,控制器由 Action 来实现,视图由一组 JSP 文件构成。因此本项目大致实现了模型、控制器、视图三者的分离,使得整个系统结构清晰。系统功能模块的实现过程可由 Struts2 的工作过程来描述:开始由用户通过浏览器发送请求,Action 控制器收到用户请求后,返回响应,调用 Action 实例的 execute()方法,由 execute()方法调用模型和 DAO 对象,实现业务逻辑,执行完后 execute()返回响应,最后再查找响应,其中,FilterDispatcher 根据配置查找响应的对应信息,诸如:success、error,将跳转到那个 JSP 页面。下面将详细介绍系统各功能模块的具体实现。

1. 用户管理

用户以浏览器为媒介发出请求信息,系统调用 struts.xml 中配置的跳转页面,验证当前用户是否具有权限,拥有何种权限。若通过后台数据库验证,则跳转到对应页面,否则停留在登录页面。struts.xml 中的配置如下:

```
    <action name="login" class="com.sanqing.action.LoginAction">
    <result name="studentSuccess"
type="chain">getRandomSubject</result><!-- 进入考试页面 -->
    <result name="teacherSuccess"
type="redirect">/teacher/index.html</result><!-- 老师登录成功页面 -->
    <result name="input">/login.jsp</result><!-- 登录失败页面 -->
    </action>
```

2. 考试功能

学生用户登录成功之后,系统会随机生成一张试卷,并保证相同考生不会出现雷同试卷。与此同时,系统开始倒计时。生成随机试题的代码如下:

```
public class GetRandomSubject extends ActionSupport
{
  private SubjectService subjectService = new SubjectServiceImpl();
  public String execute() throws Exception
  {
    List<Subject> subjects = subjectService.randomFindSubject(20);
                                                            //获得试题记录
```

```
        HttpServletRequest request = ServletActionContext.getRequest();
        request.setAttribute("subjects", subjects);
        return success;
    }
}
```

只要学生用户单击"提交"按钮或者达到规定答题时间，系统将会跳转到显示成绩页面。显示学生成绩的代码如下：

```
public class SubmitExamAction extends ActionSupport
{
  private List<Integer> subjectID;
  private SubjectService subjectService = new SubjectServiceImpl();
  private StudentService studentService = new StudentServiceImpl();
    public List<Integer> getSubjectID()
    {
        return subjectID;
    }
    public void setSubjectID(List<Integer> subjectID) {
        this.subjectID = subjectID;
    }
    public String execute() throws Exception {
        HttpServletRequest request = ServletActionContext.getRequest();
        List<String> studentAnswers = new ArrayList<String>();
        for(int i = 0; i < 20; i++) {
            String answer = request.getParameter("subjectAnswer" + i);
            studentAnswers.add(answer);
        }
        int GeneralPoint = subjectService.accountResult(subjectID, studentAnswers);
        Map session = ActionContext.getContext().getSession();
        Student student = (Student)session.get("studentInfo");
        String studentID = student.getStudentID();
        studentService.setStudentResult(studentID, GeneralPoint);
        request.setAttribute("studentName", student.getStudentName());
        request.setAttribute("GeneralPoint", GeneralPoint);
        session.put("subjectIDs", subjectID);
        return success;
    }
}
```

得到成绩之后，学生可以及时查看答案解析，相应的代码如下：

```
public class ShowSubjectAnswer extends ActionSupport{
    private SubjectService subjectService = new SubjectServiceImpl();
    public String execute() throws Exception {
        List<Subject> subjects = new ArrayList<Subject>();
        HttpServletRequest request = ServletActionContext.getRequest();
        Map session = ActionContext.getContext().getSession();
        List<Integer> subjectIDs = (List<Integer>) session.get("subjectIDs");
        for(Integer subjectID : subjectIDs) {
```

```
Subject subject = subjectService.showSubjectParticular(subjectID);
                subjects.add(subject);
            }
            request.setAttribute("subjects", subjects);
            return success;
        }
}
```

3. 教师管理

教师在本系统中设置为管理员，主要功能如下：试题、学生信息的增删改查。

（1）试题管理模块。以下是增加试题的代码，修改、删除、查看功能类似。

```
public class SubjectAddAction extends ActionSupport{
    private String subjectTitle;
    private String subjectOptionA;
    private String subjectOptionB;
    private String subjectOptionC;
    private String subjectOptionD;
    private String subjectAnswer;
    private String subjectParse;
    private int subjectType;
    private SubjectService subjectService = new SubjectServiceImpl();
    public String getSubjectTitle() {
        return subjectTitle;
    }
    public void setSubjectTitle(String subjectTitle) {
        this.subjectTitle = subjectTitle;
    }
    public String getSubjectOptionA() {
        return subjectOptionA;
    }
    public void setSubjectOptionA(String subjectOptionA) {
        this.subjectOptionA = subjectOptionA;
    }
    public String getSubjectOptionB() {
        return subjectOptionB;
    }
    public void setSubjectOptionB(String subjectOptionB) {
        this.subjectOptionB = subjectOptionB;
    }
    public String getSubjectOptionC() {
        return subjectOptionC;
    }
    public void setSubjectOptionC(String subjectOptionC) {
        this.subjectOptionC = subjectOptionC;
    }
    public String getSubjectOptionD() {
        return subjectOptionD;
    }
    public void setSubjectOptionD(String subjectOptionD) {
        this.subjectOptionD = subjectOptionD;
```

```
    }
    public String getSubjectAnswer() {
        return subjectAnswer;
    }
    public void setSubjectAnswer(String subjectAnswer) {
        this.subjectAnswer = subjectAnswer;
    }
    public String getSubjectParse() {
        return subjectParse;
    }
    public void setSubjectParse(String subjectParse) {
        this.subjectParse = subjectParse;
    }
    public int getSubjectType() {
        return subjectType;
    }
    public void setSubjectType(int subjectType) {
        this.subjectType = subjectType;
    }
    public String execute() throws Exception {
        Subject subject = new Subject();
        subject.setSubjectTitle(subjectTitle);
        subject.setSubjectOptionA(subjectOptionA);
        subject.setSubjectOptionB(subjectOptionB);
        subject.setSubjectOptionC(subjectOptionC);
        subject.setSubjectOptionD(subjectOptionD);
        subject.setSubjectAnswer(subjectAnswer);
        subject.setSubjectParse(subjectParse);
        subject.setSubjectType(subjectType);
        if(subjectService.saveSubject(subject)) {
            return success;
        }else {
            this.addActionError("该试题已经添加过了,请不要重复添加!");
            return input;
        }
    }
}
```

（2）查看成绩模块。以下是教师通过学生姓名查看成绩的代码：

```
public class QueryStudentByName extends ActionSupport{
    private String studentName;
    private StudentService studentService = new StudentServiceImpl();
    public String getStudentName() {
        return studentName;
    }
    public void setStudentName(String studentName) {
        this.studentName = studentName;
    }
    public String execute() throws Exception {
        HttpServletRequest request = ServletActionContext.getRequest();
```

```
            List<Student> students = studentService.getStudentByName(studentName);
            request.setAttribute("students", students);
            return this.success;
        }
    }
```

15.4 在线考试管理系统的运行与测试

本系统是基于 JSP 的在线考试系统，主要完成了教师增删改查学生信息、学生成绩及学生考试、系统改卷、试卷自动生成基本功能，能够基本实现客观题的答题和改卷的功能。

15.4.1 教师用户的功能运行

用户通过 login.jsp 页面输入用户名和密码，单击“登录”按钮就可以提交用户名和密码，登录界面如图 15-11 和图 15-12 所示。



图 15-11 教师用户登录界面

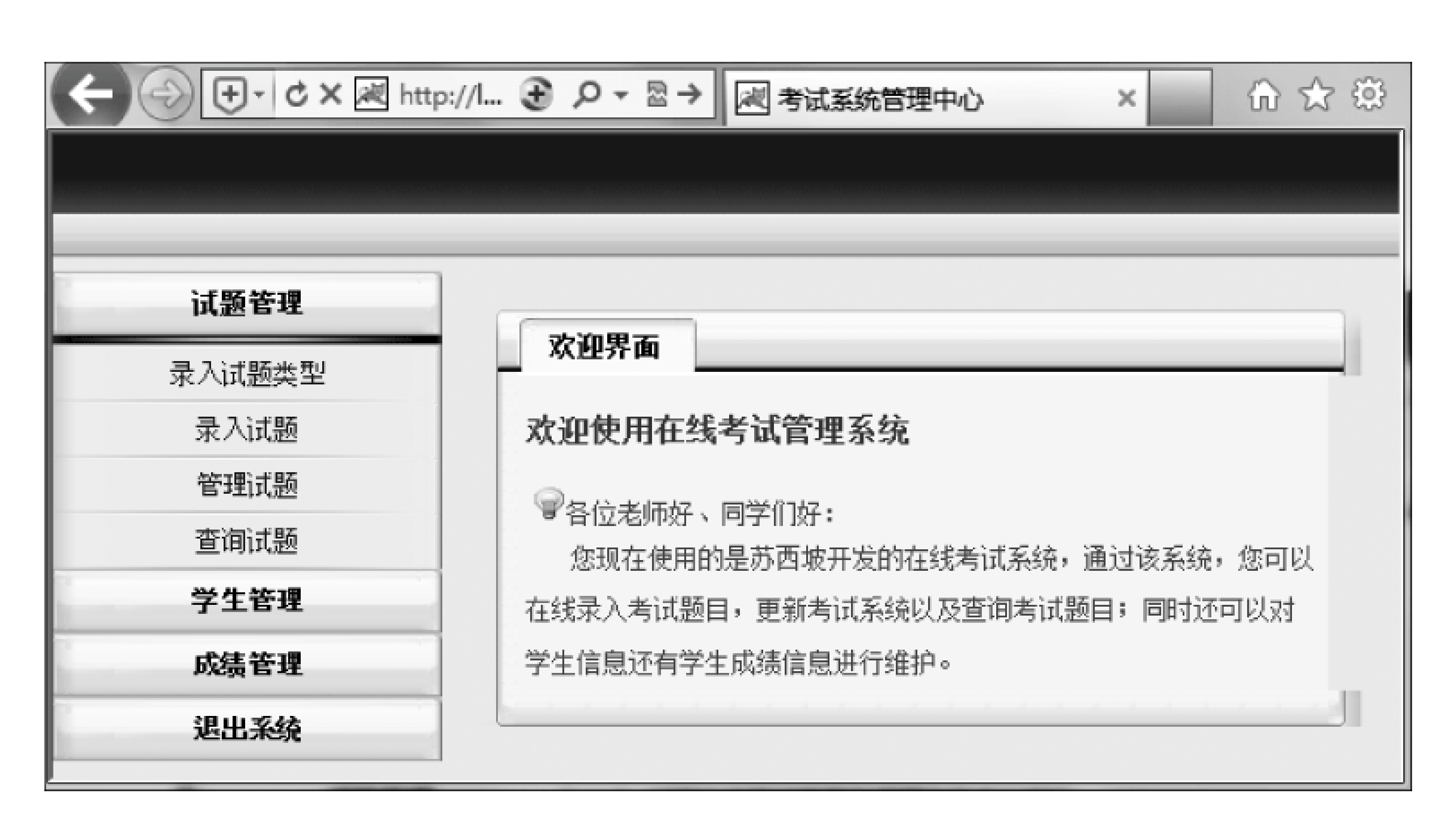


图 15-12 教师用户成功登录首页

教师管理界面中录入试题类型(科目)界面如图 15-13 所示。

教师管理界面中录入试题界面如图 15-14 所示。

在管理试题界面中，教师可进行试题的增删改查，如图 15-15 所示。

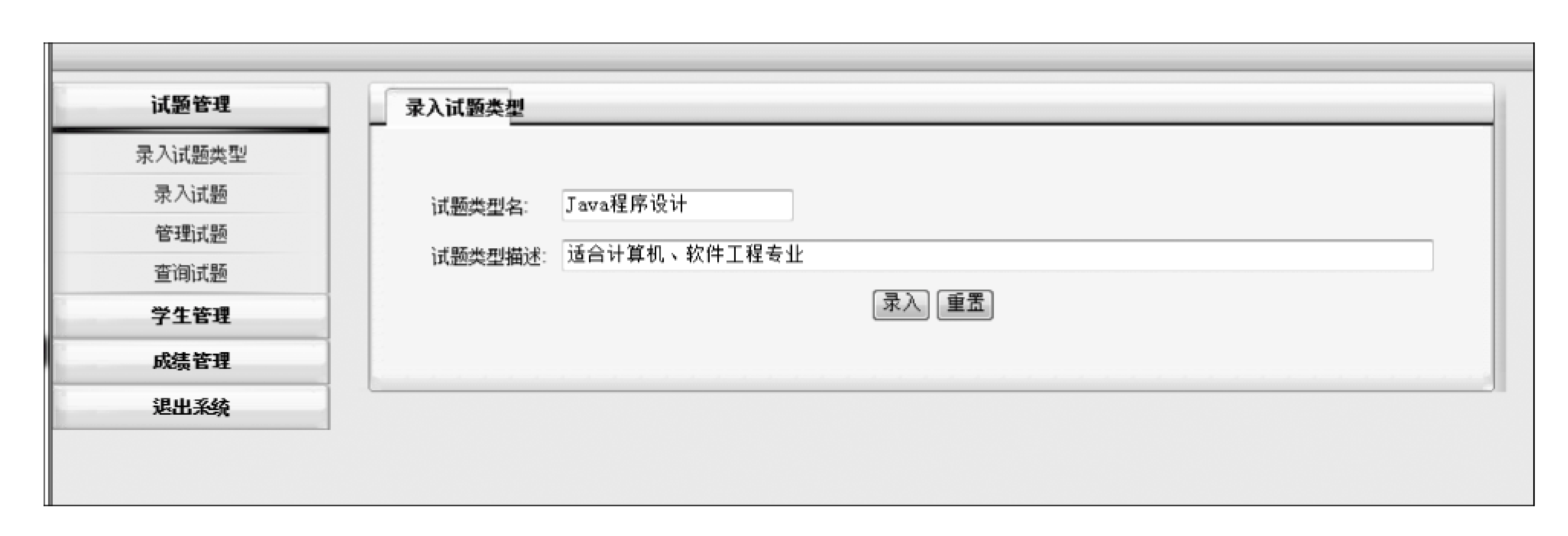


图 15-13　录入试题类型界面



图 15-14　考试管理系统录入试题界面

管理试题

试题类型	试题编号	试题标题	正确答案	查看试题	更新试题	删除试题
二级C++	6	()菜单中含有设置字体的命令。	A	查看	更新	删除
计算机基础	8	()的功能是将计算机外部的信息送入计算机。	A	查看	更新	删除
计算机基础	9	()的主要功能是使用户的计算机与远程主机相连，从而成为远程主机的终端。	C	查看	更新	删除
计算机基础	10	()视图方式可对文档不进行分页处理。	B	查看	更新	删除
计算机基础	12	()是微型计算机的外存。	C	查看	更新	删除
计算机基础	13	()是用来存储程序及数据的装置。	B	查看	更新	删除
计算机基础	14	NOVELLNETWARE是()	A	查看	更新	删除
计算机基础	15	预防计算机病毒的手段，错误的是()。	D	查看	更新	删除
计算机基础	16	“32位微型计算机”中的32指的是()	D	查看	更新	删除
计算机基础	17	“奔腾”微型计算机采用的微处理器的型号是()	D	查看	更新	删除

共31条纪录，当前第1/4页，每页10条纪录　首页　|上一页　|下一页　|尾页

图 15-15　考试管理系统试题管理界面

教师管理界面中查询试题界面如图 15-16 所示。例如,包含关键字“计算机”的查询结果界面,如图 15-17 所示。

查看试题

试题题目: 计算机

查询 重置

图 15-16 考试管理系统试题查询界面

管理试题

试题类型	试题编号	试题标题	正确答案	查看试题	更新试题	删除试题
计算机基础	8	()的功能是将计算机外部的信息送入计算机。	A	查看	更新	删除
计算机基础	9	()的主要功能是使用户的计算机与远程主机相连,从而成为远程主机的终端。	C	查看	更新	删除
计算机基础	12	()是微型计算机的外存。	C	查看	更新	删除
计算机基础	15	预防计算机病毒的手段,错误的是()。	D	查看	更新	删除
计算机基础	16	“32位微型计算机”中的32指的是()	D	查看	更新	删除
计算机基础	17	“奔腾”微型计算机采用的微处理器的型号是()	D	查看	更新	删除
计算机基础	21	“溢出”一般是指计算机在运算过程中产生的()。	C	查看	更新	删除
计算机基础	22	《计算机软件条例》中所称的计算机软件(简称软件)是指()。	D	查看	更新	删除

共8条纪录,当前第1/1页,每页10条纪录 首页 |上一页 |下一页 |尾页

图 15-17 考试管理系统查询试题结果界面

在学生信息管理界面中,教师可对学生信息进行增删改查,如图 15-18 所示。



学生编号	所属班级	学生姓名	增加学生	更新学生信息	删除学生信息
123456789	软件1504	招望舒	增加	更新	删除
123457899	计科1502	李明华	增加	更新	删除
070408000	软件1505	徐艮	增加	更新	删除
081523000	软件1502	张思香	增加	更新	删除
091578900	软件1502	王法务	增加	更新	删除

图 15-18 考试管理系统学生信息管理界面

教师管理界面中查询所有学生所有科目的成绩界面如图 15-19 所示。

教师管理界面中查询所有学生某一科目的成绩界面如图 15-20 所示。



图 15-19　考试管理系统查询班级成绩界面



图 15-20　考试管理系统查询某科成绩的界面

15.4.2　学生用户的功能运行

学生成功登录考试系统后，进入考试科目选择界面，如图 15-21 所示。



图 15-21　考试管理系统学生选择考试科目界面

学生选好科目后，进入考试界面，如图 15-22 所示。

提交试卷界面如图 15-23 所示。

考生查看答案解析界面如图 15-24 所示。

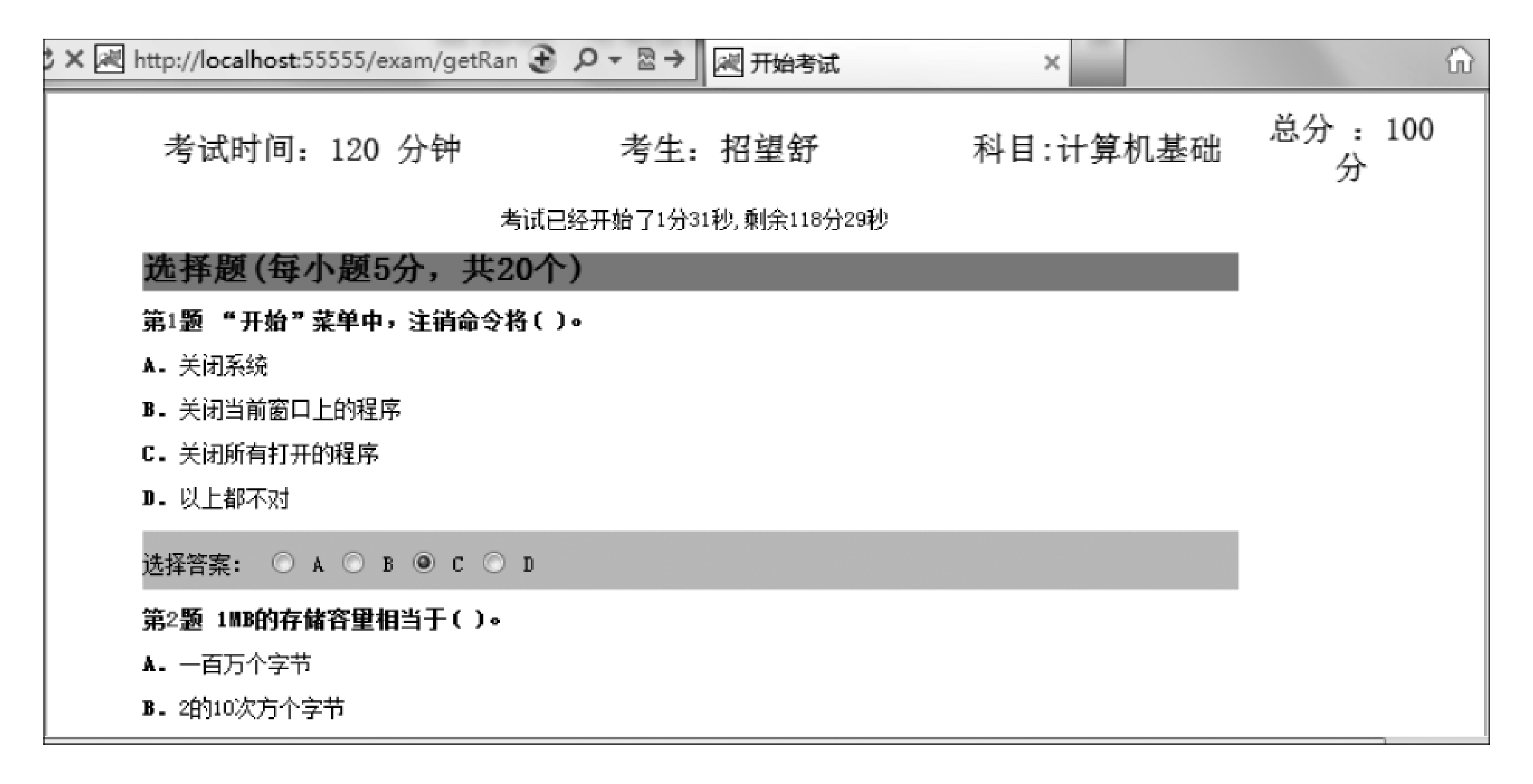


图 15-22　考试管理系统学生考试界面



图 15-23　提交试卷界面

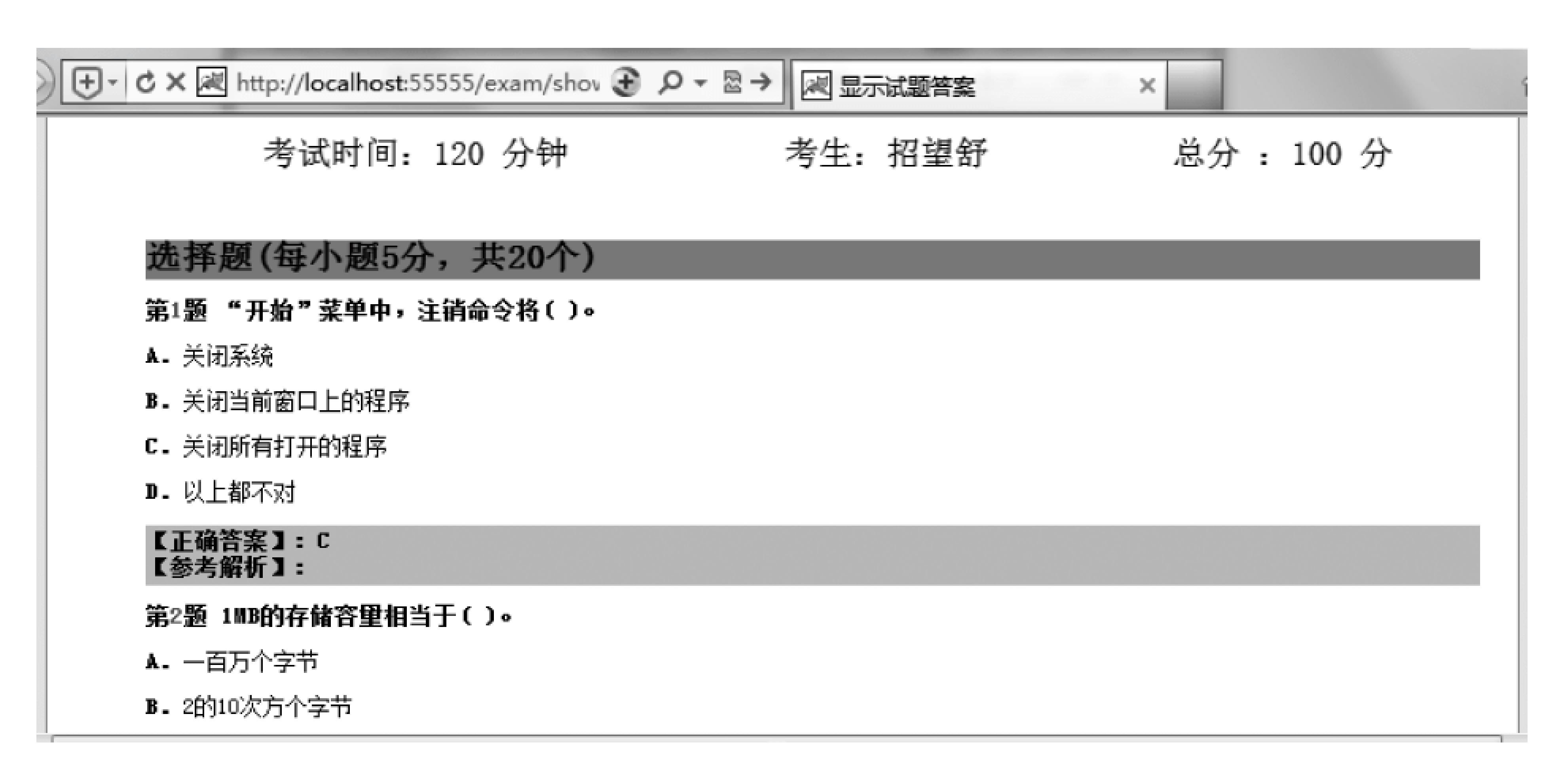


图 15-24　考生查看答案解析界面

基于 JSP 的在线系统能够帮助老师解决一些关于考试的问题，但就目前来看，系统还有很大的改进空间，最重要的是系统结构的改进，也是日后可能成为系统安全方面瓶颈的问

题。除此之外，实现技术也可以进行改进，将服务放在不同的服务器上，由特定的负载均衡算法进行调度。

15.5 小　　结

本章以基于 JSP 的在线考试系统的设计为例，介绍了利用 JSP 技术访问 MySQL 数据库的过程和技术要点。因为本书主要是介绍 MySQL 数据库的使用，所以数据库设计部分结合了本书前面介绍的知识点。在数据库设计部分，不仅涉及了表和字段的设计，还涉及了其他数据库对象等内容。其中，为了提高表的查询速度，有意识地在表中增加了冗余字段，这是数据库的性能优化的内容。在系统实现部分，需要读者对 Java 语言和 J2EE 有相应的了解。通过本章的学习，希望读者对项目开发中如何使用 MySQL 数据库有一个全新的认识。

习　题　15

1. 选择题

(1) 在关系数据库设计中，设计关系模式属于数据库设计的________阶段。

A. 需求分析　　B. 概念设计　　C. 逻辑设计　　D. 物理设计

(2) E-R 图提供了表示信息世界中实体、属性和________的方法。

A. 数据　　B. 联系　　C. 表　　D. 模式

(3) E-R 图是数据库设计的工具之一，它一般适用于建立数据库的________。

A. 逻辑模型　　B. 结构模型　　C. 物理模型　　D. 概念模型

(4) 将 E-R 图转换到关系模式时，实体与联系都可以表示成________。

A. 属性　　B. 关系　　C. 键　　D. 域

(5) 如果关系模式 R 属于 1NF，且每个非主属性都完全函数依赖于 R 的主码，则 R 属于________。

A. 2NF　　B. 3NF　　C. BCNF　　D. 4NF

2. 思考题

(1) 简述 MySQL 数据库设计优化的基本过程。

(2) 简述对于系统功能进行分析的基本要求。

(3) 说明关系模型的基本术语的含义。

题。除此之外，实现技术也可以进行改进，将服务放在不同的服务器上，由特定的负载均衡算法进行调度。

15.5 小　　结

本章以基于JSP的在线考试系统的设计为例，介绍了利用JSP技术访问MySQL数据库的过程和技术要点。因为本书主要是介绍MySQL数据库的使用，所以数据库设计部分结合了本书前面介绍的知识点。在数据库设计部分，不仅涉及了表和字段的设计，还涉及了其他数据库对象等内容。其中，为了提高表的查询速度，有意识地在表中增加了冗余字段，这是数据库的性能优化的内容。在系统实现部分，需要读者对Java语言和J2EE有相应的了解。通过本章的学习，希望读者对项目开发中如何使用MySQL数据库有一个全新的认识。

习　题　15

1. 选择题

(1) 在关系数据库设计中，设计关系模式属于数据库设计的________阶段。

A. 需求分析　　B. 概念设计　　C. 逻辑设计　　D. 物理设计

(2) E-R图提供了表示信息世界中实体、属性和________的方法。

A. 数据　　B. 联系　　C. 表　　D. 模式

(3) E-R图是数据库设计的工具之一，它一般适用于建立数据库的________。

A. 逻辑模型　　B. 结构模型　　C. 物理模型　　D. 概念模型

(4) 将E-R图转换到关系模式时，实体与联系都可以表示成________。

A. 属性　　B. 关系　　C. 键　　D. 域

(5) 如果关系模式R属于1NF，且每个非主属性都完全函数依赖于R的主码，则R属于________。

A. 2NF　　B. 3NF　　C. BCNF　　D. 4NF

2. 思考题

(1) 简述MySQL数据库设计优化的基本过程。

(2) 简述对于系统功能进行分析的基本要求。

(3) 说明关系模型的基本术语的含义。

题。除此之外，实现技术也可以进行改进，将服务放在不同的服务器上，由特定的负载均衡算法进行调度。

15.5 小　　结

本章以基于JSP的在线考试系统的设计为例，介绍了利用JSP技术访问MySQL数据库的过程和技术要点。因为本书主要是介绍MySQL数据库的使用，所以数据库设计部分结合了本书前面介绍的知识点。在数据库设计部分，不仅涉及了表和字段的设计，还涉及了其他数据库对象等内容。其中，为了提高表的查询速度，有意识地在表中增加了冗余字段，这是数据库的性能优化的内容。在系统实现部分，需要读者对Java语言和J2EE有相应的了解。通过本章的学习，希望读者对项目开发中如何使用MySQL数据库有一个全新的认识。

习　题　15

1. 选择题

(1) 在关系数据库设计中，设计关系模式属于数据库设计的________阶段。

A. 需求分析　　B. 概念设计　　C. 逻辑设计　　D. 物理设计

(2) E-R图提供了表示信息世界中实体、属性和________的方法。

A. 数据　　B. 联系　　C. 表　　D. 模式

(3) E-R图是数据库设计的工具之一，它一般适用于建立数据库的________。

A. 逻辑模型　　B. 结构模型　　C. 物理模型　　D. 概念模型

(4) 将E-R图转换到关系模式时，实体与联系都可以表示成________。

A. 属性　　B. 关系　　C. 键　　D. 域

(5) 如果关系模式R属于1NF，且每个非主属性都完全函数依赖于R的主码，则R属于________。

A. 2NF　　B. 3NF　　C. BCNF　　D. 4NF

2. 思考题

(1) 简述MySQL数据库设计优化的基本过程。

(2) 简述对于系统功能进行分析的基本要求。

(3) 说明关系模型的基本术语的含义。

图 书 资 源 支 持

感谢您一直以来对清华版图书的支持和爱护。为了配合本书的使用，本书提供配套的资源，有需求的读者请扫描下方的“书圈”微信公众号二维码，在图书专区下载，也可以拨打电话或发送电子邮件咨询。

如果您在使用本书的过程中遇到了什么问题，或者有相关图书出版计划，也请您发邮件告诉我们，以便我们更好地为您服务。

我们的联系方式：

地　　址：北京海淀区双清路学研大厦 A 座 707

邮　　编：100084

电　　话：010－62770175－4604

资源下载：http://www.tup.com.cn

电子邮件：weijj@tup.tsinghua.edu.cn

QQ：883604（请写明您的单位和姓名）



书圈

用微信扫一扫右边的二维码，即可关注清华大学出版社公众号“书圈”。